UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉNICA SUPERIOR

Departamento de Teoría de la Señal y Comunicaciones

Grado de Ingeniería de Sistemas de Comunicaciones

Trabajo de Fin de Grado

Pruebas de Transmisión con equipos software radio

AUTORA: María Martín Fernández

TUTOR: Dr. Víctor Pedro Gil Jiménez

Leganés, Julio de 2012

Título: Pruebas de Transmisión con equipos software radio.

Autora: María Martín Fernández
Tutor: Dr. Víctor Pedro Gil Jiménez

EL TRIBUNAL

Presidente: José Luis Vázquez Roig
Vocal: Miguel Lázaro Gredilla
Secretario: José Antonio García Souto

Realizado el acto de defensa y lectura del trabajo de fin de grado el día 4 de Julio de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la calificación de

PRESIDENTE VOCAL SECRETARIO

Agradecimientos

Con mucho amor a mis padres por haber confiado siempre en mí, por su apoyo y su cariño. Sin ellos no sería la persona que soy ahora. A mi madre, por estar siempre conmigo y haberme ayudado en todo, por distraerme cuando estaba estudiando, por ponerse incluso más nerviosa que yo en los exámenes, ya que como ella dice, ella también se merece el título. A mi padre que con mucho esfuerzo y trabajo siempre me lo ha dado todo.

A Pablo por estar ahí siempre cuando le necesito, por cada día de apoyo y por esos momentos de alegrías.

A mi tía Dolores por sus vitaminas para que no me fallaran las fuerzas en este largo periodo.

A Víctor Gil por darme la oportunidad de poder hacer este trabajo.

A Kun por ser un gran compañero de laboratorio y mejor amigo.

A todos los amigos que he ido haciendo durante este largo periodo, en especial al grupo del “rincón”, por esos buenos momentos de risas, de fiestas y también de desesperación.

Gracias a todos por haber hecho que este periodo de universidad sea la etapa más feliz de mi vida y que nunca olvidaré.

Resumen

Este trabajo quiere dar una idea general para utilizar el dispositivo Lyrtech SFF SDR en los sucesivos trabajos que se realicen con él. Bajo unas explicaciones teóricas previas se pretende llegar a conseguir programar una señal para explorar las posibilidades y establecer un marco de referencia para su utilización.

Además de eso, se pretende estudiar las tecnologías basadas en procesadores digitales de señales (DSP) y sistemas de lógica programable (FPGAs). Como resultado del estudio de estas tecnologías, además de aspectos teóricos sobre la operación y características, se hace un análisis de los fabricantes dominantes.

Este trabajo no sólo se basa en la recopilación de información, sino que también dará un ahorro considerable de trabajo a las personas que en un futuro se involucren en el diseño de un sistema de comunicaciones con este dispositivo.

Índice

Lista de Figuras

Lista de tablas

Lista de acrónimos

ADC	Analog to Digital Converter.
API	Application Programming Interface.
BSDK	Board Software Development kit.
CCS	Code Composer Studio.
DAC	Digital to Analog Converter.
DMA	Direct memory access.
DMP	Digital Media Processor.
DSP	Digital Signal Processing.
EDIF	Electronic Design Interchange Format.
FFT	Fast Fourier Transform.
FIFO	First In First Out.
FPGA	Field Programmable Gate Array.
FRS	Family Radio Service.
GMRS	General Mobile Radio Service.
GPP	General Purpose Preprocessor
IDE	Integrated Development Environent
IP	Intellectual Property
LSI	Large Scale Integration.
MBDK	Model-Based Development Kit.
NCO	Numerically Controlled Oscillator.
PLL	Phase Locked Loop.
RAM	Random-Access Memory.
SDR	Software Defined Radio.
SFF	Small Form Factor.

SoC	System on Chip.
TFG	Trabajo Final de Grado
VCO	Voltage-Controlled Oscillator.
VHDL	VHSIC Hardware Description Language.
VLSI	Very Large Scale Integration.
VPBE	Video Processing Back End.
VPFE	Video Processing Front End.
VPSS	Video Processing Subsystem.

1 INTRODUCCIÓN

Motivación

En nuestros días hay un gran avance en la tecnología. Las comunicaciones inalámbricas se han convertido en una herramienta imprescindible en la sociedad en la que vivimos. Día a día se trabaja en conseguir mejorar las prestaciones de estos sistemas inalámbricos y así poder afrontar nuevas necesidades.

Como consecuencia de este desarrollo, el estudio e implementación hardware de sistemas de comunicaciones inalámbricas tiene una gran proyección en el futuro de las telecomunicaciones. Los desarrolladores están considerando el uso de procesadores de señal digital (DSP, Digital Signal Processing), combinada con sistemas de lógica programable (FPGAs, Field Programmable Gate Array) para lograr un margen suficiente de procesamiento. Las plataformas de desarrollo basadas en FPGAs han experimentado una gran evolución en los últimos años gracias a la mejora de la velocidad de proceso y capacidad de almacenamiento.

El dispositivo SFF SDR (Small Form Factor Software Defined Radio) de Lyrtech es una placa Software Radio que permite enviar y recibir señales. El objetivo de esta tecnología es llevar el software lo más cerca posible de la antena, convirtiendo así los problemas de hardware en el software. Los precios bastante razonables, su tamaño compacto y sus buenas características, hacen que esta placa sea un dispositivo muy adecuado para las comunicaciones militares, seguridad pública y comercial.

La motivación para este TFG (Trabajo Final de Grado) fue conseguir familiarizarse con esta tecnología, explorar las posibilidades y establecer un marco de referencia para su utilización. Los estudiantes que emprenden proyectos con este dispositivo deben dedicar un largo tiempo a familiarizarse con la placa y los programas que se utilizan para su programación. Entre las dificultades que deben encarar está el hecho de que la información es insuficiente, o se encuentra diseminada en una serie de documentos como manuales o notas de aplicación.

Experiencias previas

En los últimos años, en el departamento de Teoría de la Señal y Comunicaciones de la Universidad Carlos III de Madrid, se han llevado a cabo diferentes estudios con el dispositivo SFF SDR de Lyrtech. En la bibliografía, podemos ver en [8] y [9].

En estos trabajos se han implementado sistemas de comunicaciones, en los que se utilizaban prácticamente la totalidad de los recursos de la placa. Pero en ninguno se explica la familiarización con el dispositivo, el uso de programas con los que se trabaja y las posibilidades que ofrece. Todo esto conlleva a que cada vez que una persona va a trabajar con el dispositivo debe dedicar bastante tiempo a la familiarización.

Objetivos

El objetivo de este trabajo es ofrecer una guía básica del funcionamiento y la programación de la plataforma SFF SDR de Lyrtech.

Este TFG cuenta con una serie de objetivos que podría enumerar en:

- Familiarización de la plataforma de desarrollo SFF SDR.
- Configurar el dispositivo SFF SDR para conseguir la transmisión de una señal.
- Configurar el DSP para enviar señales en las bandas que se desee.
- Comprobar que las características que viene en el manual de usuario son las correctas, si las bandas de transmisión son las citadas, midiendo y analizando la salida del sistema transmisor en un laboratorio de comunicaciones.
- Realizar un manual de usuario para siguientes trabajos con el dispositivo.

Contenido de la memoria

Se espera que la organización del trabajo consiga ser lo más comprensible y asequible para el lector o consultor.

Capítulo 1: Se realizará una breve introducción sobre el dispositivo SFF SDR de Lyrtech.

Capítulo 2: Se hace un pequeño estudio de las tecnologías basadas en procesadores digitales de señales (DSP) y sistemas de lógica programable (FPGAs).

Capítulo 3: Se hace una detallada descripción del entorno de trabajo utilizado en este trabajo.

Capítulo 4: Se explica cómo se ha probado en dispositivo y se analizan los resultados obtenidos.

Capítulo 5: Conclusión del trabajo.

Capítulo 6: Se detallan las fases del proyecto, explicando las tareas realizadas y los principales problemas.

Capítulo 7: Se encuentran todos los apéndices para el uso del dispositivo además del presupuesto.

2 ESTADO DEL ARTE

El rápido desarrollo de la tecnología de circuitos integrados, empezando con la integración a larga escala (LSI, Large Scale Integration), y ahora de muy larga escala (VLSI, Very Large Scale Integration) de circuitos electrónicos, ha estimulado el desarrollo de computadoras digitales. Esto ha hecho posible construir sistemas digitales complejos, capaces de realizar funciones y tareas que normalmente eran demasiado difíciles y/o costosas con circuitos analógicos. De aquí que muchas de las tareas del procesamiento de señales, que convencionalmente se realizaban analógicamente, se realicen hoy mediante hardware digital, haciendo posible la modificación de una función programada sin modificar el funcionamiento del circuito, lo que a final de cuentas lo hace más económico, y a menudo más confiable.

Procesamiento de señales digitales

Es un área de la ciencia e ingeniería que se ha desarrollado en los últimos 20 años. Este progreso es el resultado de los avances en la tecnología de la computación digital y la fabricación de los circuitos integrados. El rápido desarrollo en la tecnología de circuitos integrados ha consolidado el desarrollo de potentes, rápidas, pequeñas y baratas computadoras digitales y el hardware digital de propósitos especiales. Estos circuitos digitales han hecho posible la construcción de avanzados y sofisticados sistemas digitales que son capaces de ejecutar funciones complejas de procesamiento de señales digitales; y tareas las cuales son muy difíciles o extensas a ser ejecutadas por circuitería analógica o sistemas de proceso de señales analógicas.

DSP

El termino DSP se aplica a cualquier chip que trabaje con señales representadas de forma digital. Son microprocesadores específicamente diseñados para realizar procesado digital de señal. Todo el diseño de un DSP se basa en el manejo eficiente de una señal digitalizada, ya que debido a la latencia típica de cualquier procesador, la operación matemática debe hacerse en un tiempo acotado, para que se pueda trabajar en tiempo real. Cuando se convierte una señal analógica en digital, ésta es muestreada cada cierto intervalo, por lo que cualquier operación matemática debe hacerse entre muestras. Un DSP proporciona una rapidez y baja latencia, que permite el procesado de señales en tiempo real, y aun así siendo una herramienta de bajo consumo.

Los DSPs utilizan arquitecturas especiales para acelerar los cálculos matemáticos. Incluyen circuitería para ejecutar de forma rápida operaciones de multiplicar y acumular, conocidas como MAC. La mayoría de los DSPs incluyen en el propio chip periféricos especiales e interfaces de entrada salida que permiten que el procesador se comunique eficientemente con el resto de componentes del sistema, tales como convertidores analógico-digitales o memoria.

La diferencia esencial entre un DSP y un microprocesador es que el DSP tiene características diseñadas para soportar tareas de altas prestaciones repetitivas y numéricamente intensas. Por el contrario, los microprocesadores de propósito general o microcontroladores no están especializados para ninguna aplicación en especial. Otra diferencia muy importante es la estructura de memoria que poseen. En un microcontrolador es posible encontrar una memoria lineal, en la que se almacenan tanto datos como instrucciones de programa. Un DSP posee dos bloques separados e independientes de memoria (arquitectura Harvard), cada uno con su propio bus de acceso, permitiendo así al procesador ir a buscar la siguiente instrucción y dato en el mismo ciclo de reloj (`Fectch`).

Una de las ventajas más importantes de los DSPs es que los sistemas digitales son más seguros que los sistemas analógicos, además de una mayor flexibilidad, precisión y exactitud. Con los sistemas digitales se puede controlar la exactitud de la señal de salida. También un DSP se puede reconfigurar fácilmente.

FPGA

Una FPGA es un circuito integrado de propósito general que es programado por el diseñador más que por el fabricante del dispositivo. Puede ser reprogramada, incluso después de que se haya integrado como parte de un sistema electrónico. Se caracterizan por disponer de una serie de elementos básicos configurables, dispuestos en filas y columnas que se pueden programar y combinar para realizar funciones específicas con más eficacia que un dispositivo de propósito general. Estos elementos se conectan entre sí y a los pines de entrada y salida del dispositivo a través de una red de conexiones, también reconfigurables.

Una FPGA es programada por la descarga de un archivo de configuración llamado `bitstream` en una memoria estática de acceso aleatorio que está contenida dentro del chip. La posibilidad de reconfigurar los elementos básicos, dota a los diseños realizados de la flexibilidad necesaria para adaptarse a un mercado que cambia con rapidez.

El aumento en la popularidad y los avances en la tecnología han permitido que los precios sean más bajos y puedan competir con otros dispositivos tradicionales.

Estudio comparativo

La enorme demanda que se ha tenido en el uso de estos dispositivos ha permitido un gran avance en su tecnología con mejoras que los posicionan en aplicaciones especiales, como la telefonía inalámbrica, aplicaciones multimedia, audio, video, medicina, sistemas de control, etc.

Como consecuencia del rápido crecimiento del mercado, los vendedores de DSP compiten en ese mercado de consumo. Así, cada uno de ellos promueve sus propios circuitos integrados y herramientas de desarrollo. Entre los principales fabricantes de DSP se encuentran Analog Devices, Lucent Technologies (antes AT&T), Freescale (antes Motorola) y Texas Instruments. En la bibliografía se puede ver en [15], [16], [17] y [4] los sitios web de cada fabricante.

El fabricante Analog Devices produce tres familias de DSP, Blanckfin, Tiger Shark y ADSP-21xx. La familia Blankfin está orientada a aplicaciones multimedia y telecomunicaciones que requieren gran capacidad de cálculo, como para procesar señales de audio y video en tiempo real, comunicaciones para acceso a redes e Internet y dispositivos portátiles en donde es de vital importancia el bajo consumo de potencia. La familia Tiger Shark incluye un nuevo procesador que adopta características de diversas arquitecturas. La familia ADSP-21xx es la primera familia de dispositivos de Analog Devices. Los procesadores se están usando en módems, audio y PC multimedia.

El fabricante Freescale produce dos familias, DSP56xxx y StarCore SC100. La familia StarCore es de punto fijo compatible en instrucciones y software binario. Su arquitectura los hace aplicables en áreas como las comunicaciones inalámbricas de 3G, bancos de módems y telefonía IP. La familia DSP56xxx está enfocada a aplicaciones que incluyen audio de bus de alta fidelidad e incluso soportan enlaces de videoconferencia.

El fabricante Lucent Technologies produce tres familias, DSP16xx, DSP16xxx, DSP32C/DSP32xx. De este fabricante como no tiene sitio Web no se ha encontrado suficiente información. La familia DSP16xx usa una arquitectura de Harvard modificada con dos espacios de dirección separados. Cada dirección tiene su propio bus de datos y bus de dirección de 16bit. La familia DSP16xxx es la última familia de DSP de punto fijo, están basados en la familia DSP16xx.

El fabricante Texas Instruments produce tres series, TMS320C2000, TMS320C5000 y TMS320C6000. De Texas Instruments existe una amplia

información de sus productos a través de su portal [4]. Además de fabricante, cuenta con herramientas de desarrollo, el más importante Code Composer Studio. La serie TMS320C2000 cuenta con la familia TMS320C24x y la familia TMS320C28x. La familia TMS320C24x es de bajo costo y están diseñados para aplicaciones de control de motores y otras aplicaciones embebidas, por lo que se les conoce como Controlador Digital de Señal basado en DSP. La familia TMS320C28x cuenta con mejores prestaciones que la anterior y además tiene un amplio grupo de periféricos integrados en un mismo encapsulado. Estos DSPs son de bajo costo y están diseñados para una amplia gama de aplicaciones de control de motores, sensado para control de velocidad, PWM y corrección del factor de potencia. La serie TMS320C5000 cuenta con la familia TMS320C54x y la familia TMS320C55x. La familia TMS320C54x esta pensada para comunicaciones inalámbricas y celulares, módems, audio, telefonía por cable y redes. La familia TMS320C55x se basa en la familia anterior pero agregan mejoras a la arquitectura del procesador y juego de instrucciones. Está pensada para aplicaciones que requieren una combinación fuerte entre rendimiento y una eficiencia en el consumo de energía. La serie TMS320C6000 cuenta con la familia TMS320C62x, la TMS320C64x y la TMS320C67x. La familia TMS320C62x está basada en una arquitectura VLIW, la que permite ejecutar hasta ocho instrucciones RISC por ciclo de reloj. La familia TMS320C67x extiende la arquitectura del TMS320C62x con soporte de aritmética en punto flotante y 64 bits de datos. Algunas de las aplicaciones de estas familias son reconocimiento de lenguaje, imagen y gráficos en 3D. La familia TMS320C64x implementa una arquitectura de memoria de tipo Harvard modificada, donde se mantienen separados los espacios de direcciones para instrucción y memoria de datos. Están especialmente diseñados para aplicaciones digitales multimedia, integran puertos seriales de audio, puertos de video, controladores DMA (Direct memory Access) multicanal, temporizadores de 32 bits, puertos serie multicanal, permitiendo aplicaciones de alto rendimiento tales como estaciones base inalámbricas, módems multilínea, equipo de diagnóstico de imagen, audio, video, sistemas de radar y sonar.

El mercado de los FPGAs se ha colocado en un estado donde hay dos productores de FPGA de propósito general que están a la cabeza del mismo, y un conjunto de otros competidores quienes se diferencian por ofrecer dispositivos de capacidades únicas. Los líderes son Xilinx y Altera. En la bibliografía [5] y [18] se pueden encontrar su sitio web. Y los otros competidores son Lattice Semiconductor, Actel, Atmel, Achronix Semiconductor, entre otros.

Igual que los fabricantes de DSPs, los fabricantes de FPGAs tienen familias de FPGAs. Las series de familias de Xilinx más importantes son la Spartan y la Virtex. La serie Virtex incluye características que incluyen FIFO (First In

First Out), bloques de MAC Ethernet y transmisores y receptores de alta velocidad. La serie Spartan dirige las aplicaciones con un impacto de baja potencia, sensibilidad y coste. De Altera las familias más importantes son la Cyclone, de bajo coste, y la Stratix de alto rendimiento.

3 ENTORNO DE TRABAJO

En este capítulo se describe el hardware y software utilizado para la realización de este TFG. Los documentos utilizados para esta descripción son los manuales de usuario que ofrece el fabricante [1] [2] [3] y [19].

Este TFG se ha realizado en el Centro Mixto EADS CASA para comunicaciones y seguridad del edificio de la universidad Carlos III de Madrid en el Parque Tecnológico de Leganés. En este laboratorio se encuentra la placa SFF SDR de Lyrtech y un osciloscopio Agilent infiniium DSO90604A en donde se medirán las señales enviadas.

Herramientas de Hardware

SFF SDR

El dispositivo que se ha utilizado es la plataforma de desarrollo SFF SDR (Small Form Factor Software-Defined-Radio) fabricada por la empresa Lyrtech.

Lyrtech desarrolla y fabrica soluciones avanzadas de procesamiento de señales digitales para las empresas de todo el mundo. Ofrece una gama completa de plataformas de desarrollo DSP-FPGA, además del diseño, prototipado y fabricación de productos electrónicos.

La plataforma SDR SFF está concebida y diseñada para el desarrollo de aplicaciones radio por software. Combina la flexibilidad de una arquitectura modular, escalable del hardware con el fácil manejo del software de diseño. Posee numerosas aplicaciones desde comerciales hasta militares, ya que tiene la capacidad de manejar diferentes protocolos con la misma base de hardware.

La plataforma SFF SDR viene con dos paquetes, uno de desarrollo software (BSDK, Board Sotfware Development Kit) y otro de desarrollo basado en modelos (MBDK, Model-Based Development Kit). El BSDK permite a los usuarios desarrollar rápidamente código C para el DSP y código HDL (Hardware Description Language) para la FPGA dando a los usuarios una comprensión de todas las interfaces principales de la plataforma como el VPSS (Video Processing Subsystem), códec de audio, el módulo de conversión de datos o el módulo de radiofrecuencia. Del mismo modo, el MBDK permite a los usuarios desarrollar aplicaciones para la plataforma con Simulink dentro de MATLAB. Al dirigirse a la DSP y FPGA con herramientas MBDK, los usuarios pueden implementar y validar algoritmos en el hardware más rápido.

A diferencia de otras plataformas de desarrollo SDR del mercado, la SFF SDR al separar la banda base, la frecuencia intermedia y la radio frecuencia de unos módulos a otros (en lugar de mantener una arquitectura única y fija), los desarrolladores pueden extender sus capacidades de desarrollo de radio y optimizar los costes y el consumo de energía.

La plataforma tiene tres placas, el módulo de procesamiento digital, módulo de conversión de datos y módulo de radiofrecuencia. (Ver figura 3.1).

El módulo de procesamiento digital está equipado de un FPGA Virtex-4 y un SoC (System on Chip) DM6446 que ofrece a los desarrolladores el rendimiento necesario para implementar funciones personalizadas con diferentes requisitos de un protocolo a otro. El módulo de conversión de datos está equipado con convertidores de doble canal analógico-digital y digital-analógico. El módulo de radiofrecuencia abarca una variedad de gamas de frecuencia en la transmisión y recepción, que le permite soportar una amplia gama de aplicaciones. A continuación se describe con más precisión las partes principales de cada uno de estos módulos.

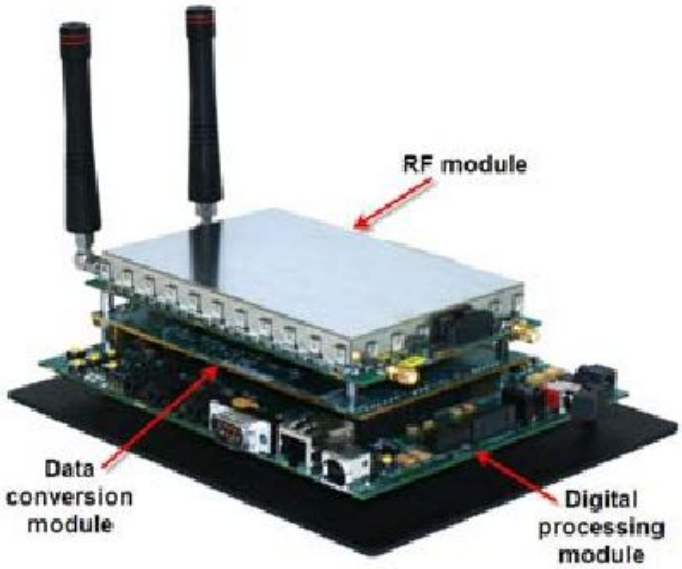

Figura 3.1: Plataforma SFF SDR.

En la figura 3.2 muestra los tres módulos de la plataforma.

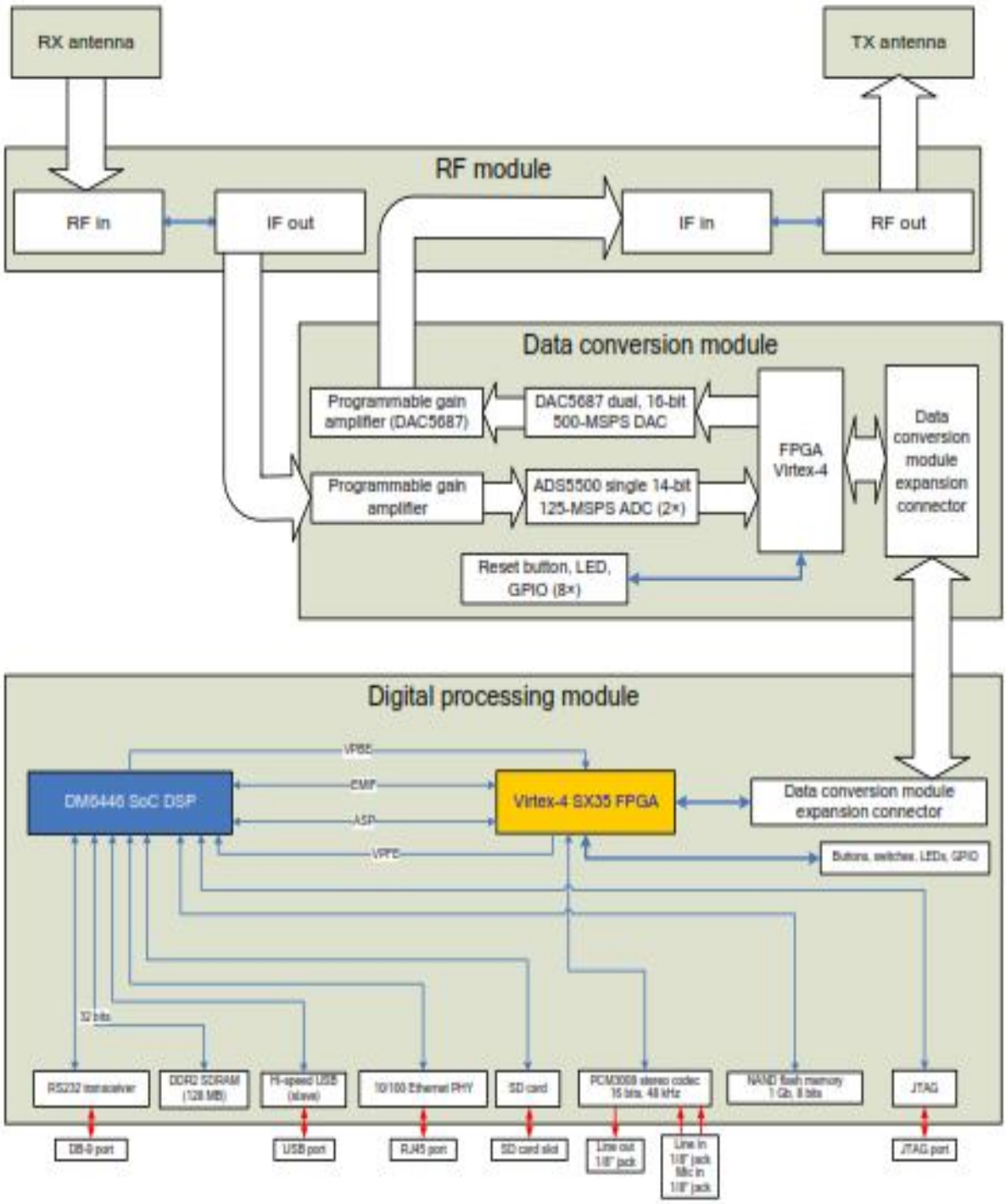

Figura 3.2: SFF SDR diagrama de bloques

Módulo de procesamiento digital

En esta sección se proporciona una descripción de las principales características del módulo de procesamiento digital. (Ver figura 3.3)

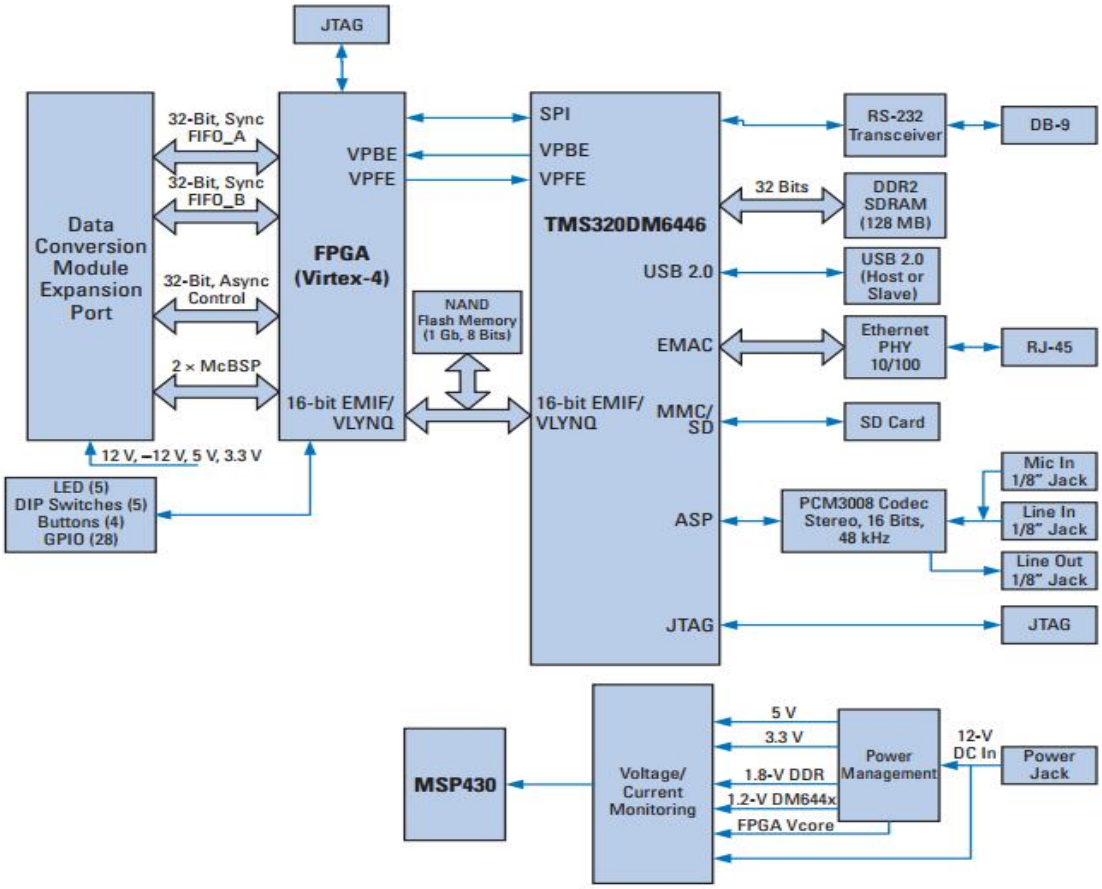

Figura 3.3: Bloques del módulo de procesamiento digital

Esta placa contiene dos componentes principales, un DSP y una FPGA:

Un procesador digital de señal

Es un chip integrado TMS320DM6446 DMP (`Digital Media Processor`) de Texas instrument, que forma parte de la familia de DSPs Da Vinci. Esta familia de DSPs es `Media Processors` basados en la tecnología de procesamiento de señales digitales. Es `System On Chip`, toda la funcionalidad del dispositivo se programa directamente sobre el mismo chip.

Este DSP está compuesto por un procesador ARM9 de propósito general (`GPP, General Purpose Preprocessor`) y un procesador TMS320C64x que es el núcleo. El procesador ARM9 trabaja a una frecuencia de 594 MHz y el núcleo a 297MHz. Además contiene algunas memorias como 64 registros de

32 bits, memoria RAM (Random-Access Memory) para datos y programas, y acceso a memoria externa.

FPGA

Es una Virtex-4 XC4SX35 de Xilinx. Tiene la función de trabajar como un coprocesador y como interfaz de entrada salida de la placa y los dispositivos que están en ella. Entre sus principales características se encuentra que tiene 34560 celdas lógicas y 3456 kilobits de RAM

Se puede llevar a cabo una comunicación entre el DSP y la FPGA a través del VPSS. El VPSS es una DM6446 de 16 bis, está compuesto por un VPFE (Video Processing Front End) y el VPBE (Video Processing Back End). El VPFE se utiliza como una interfaz de entrada para el DSP y el VPBE como una interfaz de salida desde el DSP. El VPSS se ha adaptado para ser utilizado en el módulo de procesamiento digital de la SFF SDR como una interfaz para la transferencia de datos entre la DSP y FPGA.

Módulo de conversión de datos

A continuación se proporciona una descripción de las principales características del módulo de conversión de datos. (Ver figura 3.4)

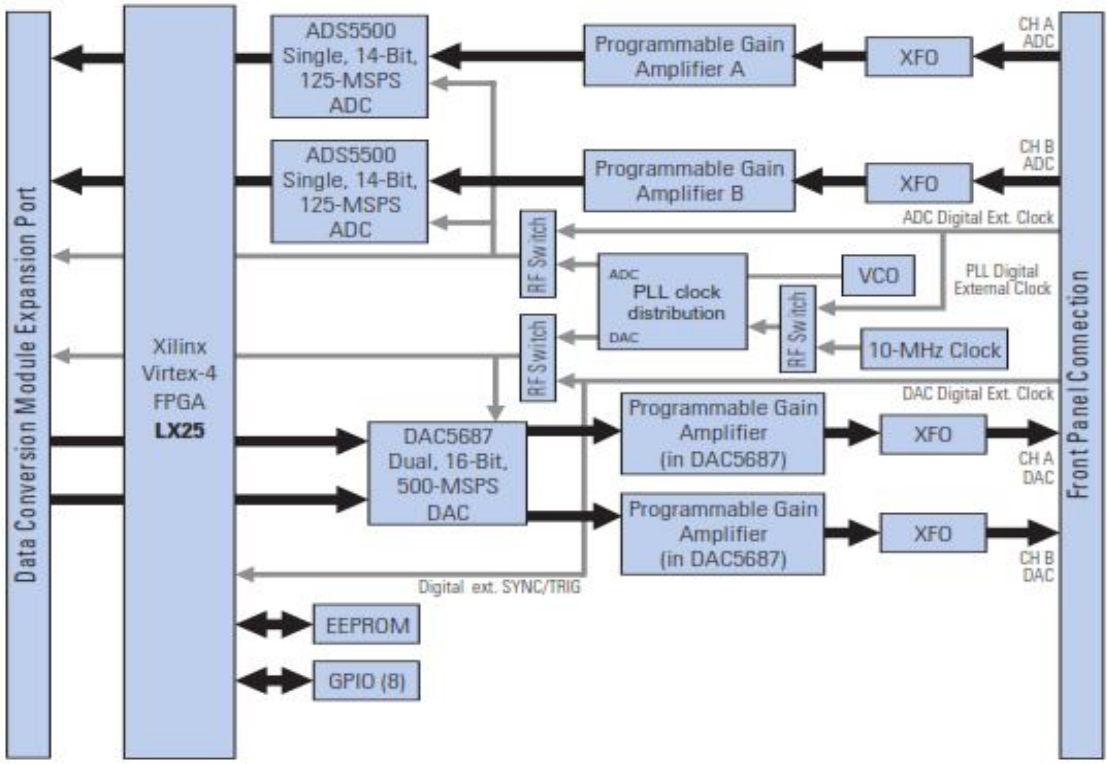

Figura 3.4: Bloques del módulo de conversión de datos

Este módulo está formado por un convertidor digital-analógico (DAC5687) y un convertidor analógico-digital (ADS5500).

El DAC5687 proporciona dos canales de salida para las señales procesadas, de 16 bits de resolución. Dispone de filtros de interpolación, un oscilador complejo controlado numéricamente (NCO, `Numerically Controlled Oscillator`) y compensación IQ. La ganancia del DAC (`Digital to Analog Converter`) se puede ajustar entre 0 dB y 20 dB. Permite diferentes modos de operación, `Dual-channel mode` y `Single-sideband mode`. En el `Dual-channel` las salidas del DAC son independientes, lo que facilita el trabajo con señales en banda base con modulación en cuadratura.

El ADS 5500 puede muestrear a una tasa de 125 Mega muestras por segundo con una resolución de 14 bits. Además, cuenta con un atenuador digital y un amplificador programable que combinándolos podemos obtener gran variedad de ganancias para las señales de entrada. El atenuador PE4305 se utiliza para disminuir la potencia de la señal de entrada hasta 15,5 dB en incrementos de 0,5 dB. El amplificador LT5514 proporciona una ganancia entre 22,5 dB y 33 dB en incrementos de 1,5 dB.

Además en este módulo hay otra FPGA, es el modelo Virtex-4 XC4VLX25 de Xilinx, se encarga de ajustar los valores de configuración del conversor de datos. Se comunica con el DSP a través del conector de expansión. Tiene 24192 celdas lógicas y 1296 kilobits de RAM. Esta FPGA no es para ser reprogramada sino es para acceder a lo parámetros de configuración a través de la DSP.

Módulo de radiofrecuencia

El módulo de radiofrecuencia se compone de una sección de RX (receptor) y otra de TX (transmisor). (Ver figura 3.5)

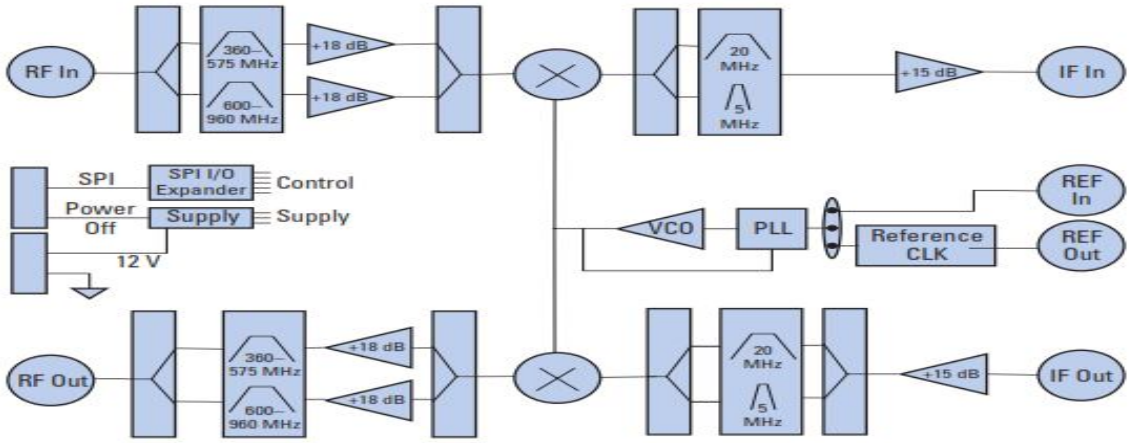

Figura 3.5: Bloques del módulo de radiofrecuencia

TRANSMISOR

En la siguiente figura se puede ver el esquema del transmisor. (Ver figura 3.6)

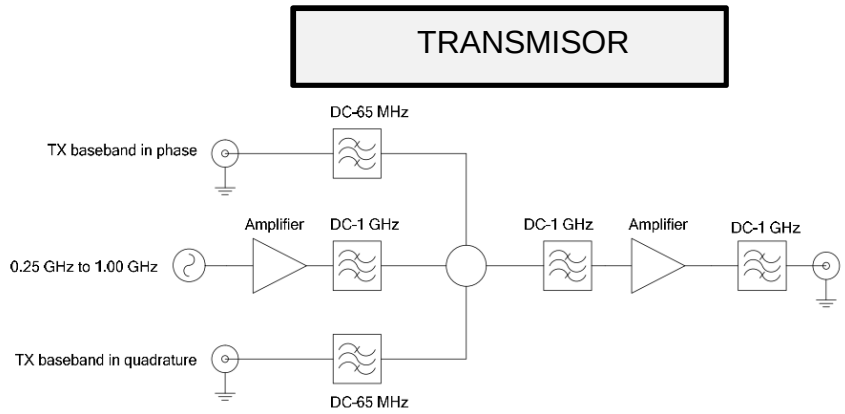

Figura 3.6: Esquema transmisor

La sección del transmisor permite la transmisión de señales en dos bandas, la primera de 262 a 438 MHz y la segunda de 523 a 876 MHz. El transmisor recibe las señales en fase y cuadratura, y las mezcla con la portadora. Independientemente, las señales de fase y cuadratura son filtradas paso bajo por un filtro de 65 MHz.

Entre sus principales componentes se encuentra:

- Un oscilador local, que se compone de un sintetizador PLL (Phase Locked Loop) TRF3750, un oscilador controlado por tensión y un pre-escalador de división por dos que pueden ser activados o desactivados dependiendo de la banda de transmisión deseada. Cuando el pre-escalador está activado se transmite en la primera banda, cuando está desactivado se transmite en la segunda banda.
- Un modulador de cuadratura (TRF3701), que se utiliza para producir una salida de banda lateral única a la salida, convierte la señal a frecuencia intermedia menor de 65 MHz ya que tiene un filtro paso bajo a la entrada.
- Un reloj de referencia, que se utiliza por el receptor y el transmisor como una referencia para asegurar que todas las frecuencias están en fase. Es posible seleccionar un reloj de referencia interno de 10 MHz o uno externo que debe ser introducido en el conector de referencia de reloj de entrada del modulo de radiofrecuencia.

RECEPTOR

En la siguiente figura se puede ver el esquema del receptor. (Ver figura 3.7)

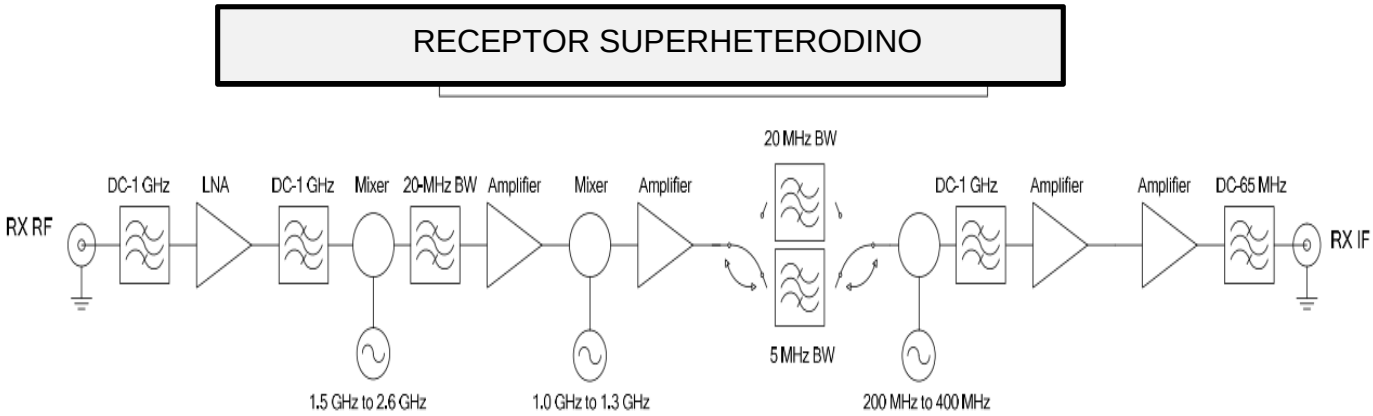

Figura 3.7: Esquema receptor

La sección del receptor está formada por un receptor superheterodino de tres etapas con una frecuencia intermedia final de 30 MHz y unos anchos de banda seleccionables de 5 MHz o 20 MHz.

Sus principales componentes son:

- Cuatro filtros utilizados para rechazar el ruido fuera de la banda de interés. El primer filtro es de un 1GHz, paso bajo. El segundo filtro es de 20 MHz, paso banda en torno a 1,575 MHz. El tercer filtro es uno seleccionable de 5MHz o 20 MHz, paso banda centrado en 300MHz. Y el último filtro es de 65 MHz, paso bajo utilizado para producir la frecuencia intermedia final enviada a ADC (`Analog to Digital Converter`) del módulo de conversión de datos.
- Tres osciladores locales utilizados en las etapas del receptor para llevar la señal de radiofrecuencia a frecuencia intermedia de modo que pueda ser procesada por el módulo de conversión de datos. Cada oscilador local se compone de un sintetizador PLL TRF3750 y se utiliza para bloquear la frecuencia del oscilador. El primer oscilador local se compone de un pre-escalador de división por dos que permite obtener frecuencias entre 1,6 y 2,5 GHz. El segundo oscilador local está fijado en 1275 MHz, mientras que el tercer oscilador se fija a 330 MHz.
- Tres mezcladores de frecuencia, uno para cada oscilador, que se utilizan para mezclar las señales de los osciladores locales con las señales recibidas para llevarlos a la frecuencia intermedia deseada.

Además hay dos antenas que trabajan en las bandas de GMRS (`General Mobile Radio Service`)/FRS (`Family Radio Service`). Tienen una ganancia de 2,5 dBi.

Herramientas de Software

En este apartado se describen las herramientas software utilizadas. Como se comentó anteriormente, la SFF SDR tiene dos modos de desarrollo, BSDK y MBDK. En este trabajo vamos a utilizar el BSDK, ya que proporciona la posibilidad de utilizar todas las funciones disponibles del sistema mediante el desarrollo de código en C para el DSP, o código HDL para la FPGA.

Para el desarrollo de las configuraciones del DSP, se ha usado el programa Code Composer Studio v3.3. Para la descripción de las configuraciones de la FPGA se ha usado el programa Xilinx ISE Foundation 12.4. Y para simular el código HDL se ha utilizado Modelsim.

Code Composer Studio v3.3

Code Composer Studio (CCS) es un IDE (`Integrated Development Environent`), es decir, un ambiente de desarrollo integrado que actúa como interface de comunicación entre la tarjeta de desarrollo y la computadora, permitiendo programar al DSP de una manera sencilla. Incorpora un compilador C, ensamblador, enlazador, simulador, depurador, etc. Con el editor permite la manipulación de programas en lenguaje C/C++ y ensamblador.

CCS permite un manejo rápido y sencillo para moverse en todas las fases del proceso de diseño de una aplicación. Facilita el uso de diversas ventanas de observación y permite establecer puntos de ruptura dentro del programa, puntos de prueba para datos de entrada/salida (I/O).

CCS incluye algunas librerías de trabajo para trabajar con diferentes versiones de plataformas de evaluación (DSP de Texas Instruments), que permiten al usuario iniciar tareas de programación. Es recomendable recurrir a los manuales de instalación de cada plataforma en particular, con el fin de ahorrar tiempo de manejo e instalación.

El compilador de C compila programas de extensión .c para luego producir un fichero de ensamblaje con extensión .asm. Una vez hecho esto el ensamblador procesa los .asm para producir objetos de lenguaje máquina con extensión .obj. El linkador combina ficheros objeto y librerías objeto como entrada para producir un fichero ejecutable con extensión .out. Este fichero ejecutable es el que finalmente puede ser cargado para ser ejecutado directamente en el DSP.

Xilinx ISE Foundation 12.4

ISE Foundation es un entorno informático compuesto por un conjunto de herramientas que asisten en el proceso de diseño, simulación, síntesis del resultado y configuración del hardware. Permite diseñar circuitos digitales por medio de esquemas lógicos, máquinas de estados o utilizando la descripción en lenguajes VHDL, Verilog. Posee todas las herramientas necesarias para la

implementación y sintetización del código HDL en la FPGA. Tiene una interfaz gráfica orientada al usuario que facilita el uso de todas sus aplicaciones por medio de menús.

Una vez hecha la descripción en lenguaje VHDL, se sintetiza y como resultado genera un fichero con extensión EDIF que contiene el listado de conexiones. Para finalizar, se implementa el diseño y se genera un fichero con extensión .bit que es el que se ejecutará directamente en la FPGA.

Modelsim

Es un sistema de simulación digital basado en VHDL y Verilog. Además de simular también permite editar, compilar y depurar diseños de sistemas digitales descritos.

Aunque Xilinx ISE posee simulador, se utiliza Modelsim debido a su interfaz gráfica de usuario de fácil manejo que permite de un modo rápido identificar y depurar los diferentes errores que puedan surgir en el funcionamiento de los circuitos. Esta interfaz muestra de forma gráfica una interpretación de todas las señales que formen parte de los diseños, pudiendo también modificar y recompilar el código para poder volver a simularlo. Agiliza la detección y depuración de errores puesto que no hace falta estar generando y volcando los modelos a la FPGA para llevar a cabo las distintas pruebas.

4 PRUEBA Y RESULTADOS DEL DISPOSITIVO SFF SDR

En este capítulo se explica cómo se ha probado el dispositivo SFF SDR de Lyrtech desde la utilización de los IP Core que ofrece la herramienta ISE de Xilinx, hasta la generación de los ejecutables para cargarlos en la placa. Además de probar el dispositivo, se ha intentado hacer un manual de usuario para futuros trabajos con esta placa. Para ello al final de este trabajo hay una sección con apéndices que intentan explicar paso a paso la utilización de los programas utilizados, la compilación, simulación y generación de los ejecutables.

En este TFG se ha implementado un seno para poder comprobar algunas de las características descritas en el manual de usuario, como, verificar las bandas de frecuencia del módulo de radiofrecuencia o comparar los valores de ganancia del DAC. Para ello se ha utilizado los manuales del fabricante y otros proyectos anteriores realizados con esta placa.

Software Lyrtech

En primer lugar, para poder realizar cualquier operación con la placa, lo primero que hace falta es instalar el software proporcionado por Lyrtech. En él se encuentran todas las librerías necesarias tanto para implementar un programa en VHDL para la FPGA como un programa en C para el DSP.

Cuando se instala el software, aparece en C:\ una carpeta llamada `Lyrtech`. En ella se encuentran dos carpetas más, `Host_SDK` y `SFF_SDR`. Las librerías que se van a utilizar se encuentran en la carpeta `SFF_SDR`. Cuando se lleve a cabo el desarrollo en VHDL de la señal se utilizarán las librerías que están en `C:\Lyrtech\SFF_SDR\SFF_SDR\fpga`. Estas librerías se usarán cuando se sintetice en Xilinx ISE el proyecto. Para que el proyecto se sintetice correctamente y se genere el ejecutable .bit hay que especificar en `Process Properties/Macro Search Path` dónde está cada librería especificando una por una todas ellas, que se encuentran dentro de la carpeta `fpga`.

Dentro de la carpeta `fpga` hay dos carpetas, `common` y `default`. En la carpeta `common` se encuentran los archivos `netlist` precompilados para las interfaces comunes de la placa y el archivo de constantes .ucf. En la carpeta `default` nos encontramos los archivos `netlist` precompilados por defecto y el código VHDL fuente.

Para programar el DSP, se utilizan otras librerías que están situadas en `C:\Lyrtech\SFF_SDR\SFF_SDR\sdk\dsp\common`. Estas librerías se

utilizan en el CCS (ver apéndice 8). Sin estas librerías no se puede crear el ejecutable .out. A parte de librerías, están todos los `include` que hacen falta para el DSP y que vienen explicados en el `API Guide` [2].

Antes de empezar a desarrollar la señal que se va a enviar, cuando se instala el Xilinx ISE (ver apéndice 3), lo inmediatamente después hay que compilar las librerías (ver apéndice 4). Estas librerías se encuentran en `C:\Xilinx\12.4\ISE\vhdl\mti_se\6.5\nt`. Aquí se localizan las carpetas `simprim`, `unisim` y `xilinxcorelib`, que son para poder simular en Modelsim.

Una vez se tenga hecho esto se puede empezar a diseñar la señal.

IP Core

Para probar el dispositivo, se manda un seno. El seno se va a generar con los IP Core que ofrece Xilinx ISE (ver apéndice 9). Los IP Core son partes de código VHDL específicos diseñados para hacer un trabajo determinado, además, permiten optimizar los recursos de la FPGA y liberar el tiempo empleado en el desarrollo de funciones estándar.

En la figura 4.1, se ve el bloque generado,

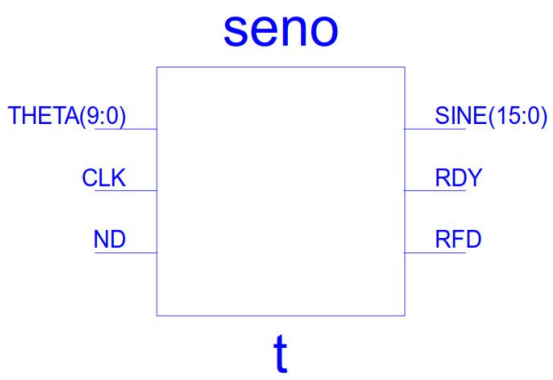

Figura 4.1: Esquema componente seno.

Los parámetros que se han configurado son:

- `THETA`: valor de entrada para el cual se genera el seno.
- `CLK`: reloj de la entrada por flaco activo.
- `ND`: indica al módulo que un nuevo valor `THETA` se está introduciendo en el puerto de entrada `THETA`. La afirmación de `ND` inicia la generación del valor de salida sinusoidal.
- `SINE`: valor de salida para una entrada de `THETA`.

- RDY: indica que un valor sinusoidal válido está disponible en los puertos de salida.
- RFD: indica que el módulo puede aceptar nuevos valores de entrada THETA.

Simulación

Para probar el módulo del seno, se simula con Modelsim (ver apéndice 6) un programa de prueba llamado `Test_seno`. Con este programa simplemente se muestra el seno generado. El resultado de la simulación se muestra en la figura 4.2.

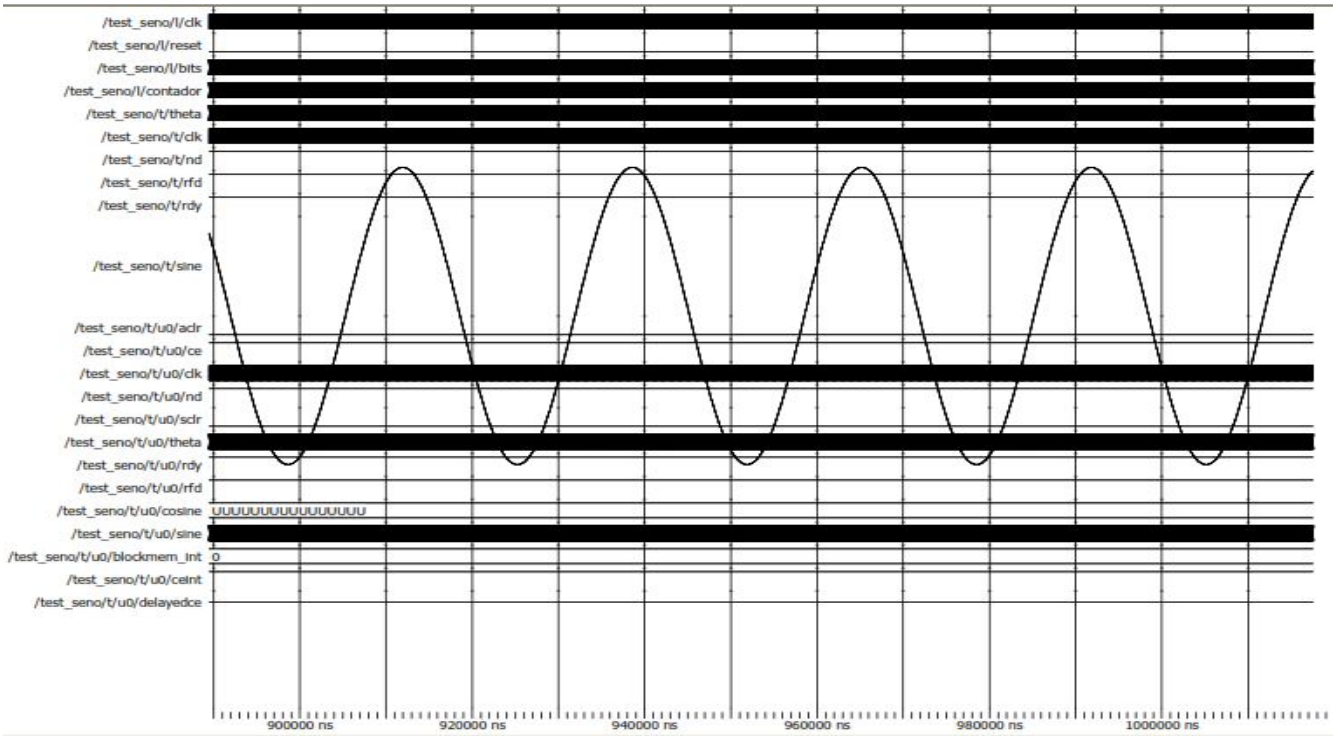

Figura 4.2: Simulación del seno.

Descripción de la señal

El dispositivo proporciona unas interfaces que sirven para desarrollar e integrar los programas. Con estas interfaces de entrada/salida, se puede trabajar con todos los módulos de los que dispone. Las interfaces se encuentran descritas en un archivo VHDL llamado `Custom_Logic`, proporcionado por Lyrtech.

El seno generado anteriormente, necesita como entrada el valor de THETA. Esa señal de THETA se genera con un contador. Al contador se le mete como entrada la señal de reloj `i_ClkMst_p` ya que va a más frecuencia que el reloj

principal de la FPGA (`i_FpgaClk_p`). El contador convierte la señal de reloj en un vector.

Como el dispositivo tiene dos buses de salida, uno del canal A (`ov16_DacDataChA_p`) y otro del canal B (`ov16_DacDataChB_p`), se saca por cada canal el mismo seno.

En resumen, las interfaces usadas en el diseño son:

- `i_Reset_p`: reset global.
- `i_ClkMst_p`: Master Clock 225MHz.
- `i_FpgaClk_p`: reloj principal de la FPGA 37.5MHz.
- `ov16_DacDataChA_p`: bus de salida al canal A.
- `ov16_DacDataChB_p`: bus de salida al canal B.
- `o_ClkCstm2Dac_p`: reloj de entrada del módulo de conversión de datos.

En la figura 4.3 se puede ver el esquema del circuito completo que será cargado en la placa.

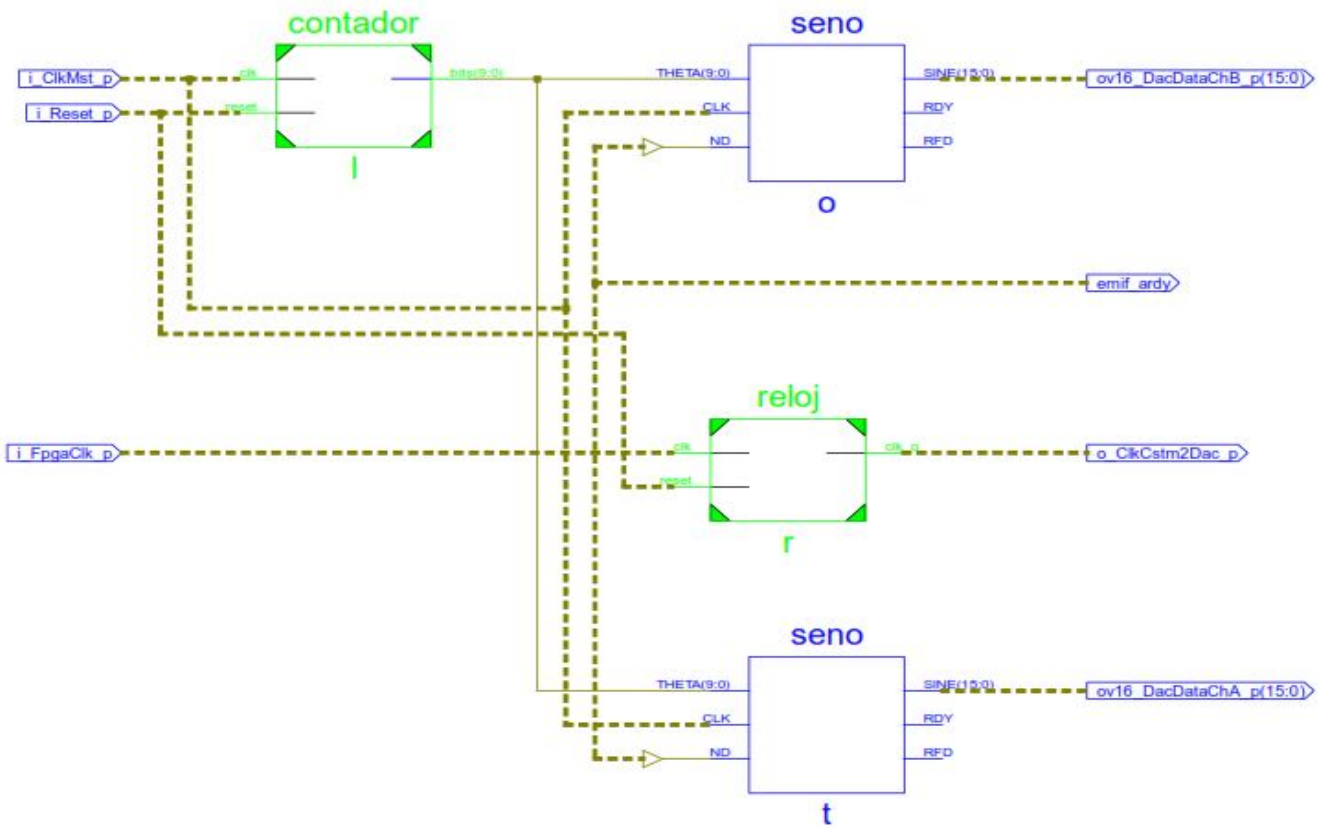

Figura 4.3: Circuito descripción de la señal.

Todo el diseño se hace en el programa ISE de Xilinx. Cuando ya se tenga todo terminado, el programa Xilinx ISE sintetiza y crea el ejecutable .bit que será cargado en la FPGA (ver apéndice 2). El programa ofrece información sobre los recursos de la placa que se han utilizado en el diseño. Es recomendable revisar esta tabla para poder optimizar el diseño. En este diseño, los resultados obtenidos se pueden ver en la figura 4.4. Como se puede ver se han empleado menos de la mitad de los recursos.

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	2,244	30,720	7%	
Number of 4 input LUTs	3,421	30,720	11%	
Number of occupied Slices	2,208	15,360	14%	
Number of Slices containing only related logic	2,208	2,208	100%	
Number of Slices containing unrelated logic	0	2,208	0%	
Total Number of 4 input LUTs	3,575	30,720	11%	
Number used as logic	3,371			
Number used as a route-thru	154			
Number used as 16x1 RAMs	16			
Number used as Shift registers	34			
Number of bonded IOBs	211	448	47%	
IOB Dual-Data Rate Flops	4			
IOB Master Pads	12			
IOB Slave Pads	12			
Number of BUFG/BUFGCTRLs	8	32	25%	
Number used as BUFGs	7			
Number used as BUFGCTRLs	1			
Number of FIFO16/RAMB16s	4	192	2%	
Number used as RAMB16s	4			
Number of DCM_ADVs	3	8	37%	
Number of RPM macros	3			
Average Fanout of Non-Clock Nets	3.57			

Figura 4.4: Tabla de resultados.

Configuración del DSP

En la configuración del DSP se configuran todos los módulos del dispositivo. Se utiliza la herramienta CCS que además genera el archivo ejecutable .out que será cargado en la placa.

En este trabajo los módulos que se van a configurar especialmente son el módulo de radiofrecuencia y el conversor de datos. El módulo de radiofrecuencia lo que va permitir es elegir la banda de frecuencia a la que se quiere transmitir. El módulo de conversor de datos va a permitir configurar los distintos valores de la ganancia del DAC.

Primeramente, hay que inicializar los módulos del dispositivo. Para ello se usan los siguientes métodos que se pueden encontrar en el `API Guide` [2]. Además de utilizar los métodos de inicialización, hay que incluir al principio del fichero las librerías que proporciona Lyrtech.

```
La inicialización de la FPGA se debe hacer antes de trabajar con ella */
 INIT_Chip();//inicializa el módulo DaVinci DM6446
 TIM_Init(); //inicializa TIMER 2
 FPGA_Init();//inicializa FPGA

 //Inicialización driver del módulo de radiofrecuencia
 RFFE_Init ();

 conv_mod_init(); //inicializa el driver del módulo de conversión de datos
 conv_mod_InitPrologue(); //ejecuta la rutina de pre-inicialización del módulo de conversión de datos
```

Seguidamente se configura el módulo de radiofrecuencia. Lo primero hay que habilitar la alimentación de los módulos de transmisión y recepción. Luego se selecciona el ancho del filtro en el receptor, que limita el ancho de banda de las señales que pueden ser recibidas. Hay dos opciones 5MHz o 20MHz. Y finalmente se configura la fuente de reloj de los osciladores. Se puede elegir entre un reloj externo y uno interno. En este este caso se usa el interno.

```
RFFE_SetPowerRX(1); // Enable RX
RFFE_SetPowerTX(1); // Enable TX

RFFE_SetRxFilter(FREQ_20MHZ); // Set bandwith to 20MHz (Options: FREQ_5MHZ or FREQ_20MHZ)
// 1: External clock
// 0: Internal clock
RFFE_SetRefClk(0);//(0: Internal clock)
```

Ahora se configuran las bandas de transmisión. Para ello hay que crearse unas variables que permitan elegir la frecuencia de transmisión y recepción de la placa. Para usar una banda u otra, hay que activar o desactivar el `prescaler`. Cuando está el `prescaler` activado, `TxFreq` se divide entre 2, y cuando no, sale directamente `TXFreq`. La variable `TXFreq` calcula la frecuencia de portadora, es decir, la frecuencia a la que vamos a transmitir.

```
//variables
float RxCarrier = 740; //in MHz
float TxCarrier = 370; //in MHz

// 0: está desactivado, usamos la 2° banda
// 1: está activado, usamos la 1° banda
RFFE_SetTxPrescaler(0); //configures the TX prescaler (1: DIV2)

// Calcula la frecuencia de TX: el canal de TX se calcula con la variable TXCarrier arriba declada
// Si el prescaler está a 1 (activado) el resultado de TXFreq está divido entre 2
TXFreq = ((int) ((float) (TxCarrier*10.0 + 0.5)) * 100000 * 2);
```

Una vez hecho esto, se configura el módulo de conversión de datos. Sólo se programará el DAC. Primero hay que activar el DAC y luego se configuran las entradas para que se interpolen de forma independiente. Las interpolaciones son x1, x2, x4 y x8, y se puede usar una corrección `sin(x)/x` en la interpolación.

```
if(conv_mod_enableDAC(ENABLE))
{
    printf("Conv module: DAC error!\n");
    return;
}

conv_mod_SetDualChanDACMode(INTERPx1, SINX_ON);
```

Para configurar los valores de la ganancia del DAC de cada uno de los canales por separado, hay dos ajustes `Coarse`, que puede tomar valores entre 0 y 15, y `Fine`, que puede tomar valores entre -128 y 127. Se cambia en el método `conv_mod_SetDACChanGain(param_cnt, channel, gain_type, gain);`

- `param_cnt`: número de canales que se va a establecer.
- `channel`: el canal que se va a establecer.
- `gain_type`: tipo de ganancia (`Coarse` o `Fine`).
- `gain`: para `Coarse` (0 a 15), para `Fine` (-128 a 127).

```
if(conv_mod_SetDACChanGain ( 4, CHAN_A, COARSE, 14, CHAN_A, FINE, 0,  CHAN_B, COARSE, 14, CHAN_B, FINE, 0))
{
    printf("Conv module: DAC error!\n");
    return;
}
```

Una vez hecho esto y generado el archivo ejecutable, se carga en la placa y se pueden ver los resultados.

Resultados

Pruebas de transmisión

Para probar el dispositivo, se cargan en la placa (ver apéndice 2) los ejecutables creados anteriormente. Se empezará midiendo la señal en banda base y luego en radiofrecuencia.

En la figura 4.5 se ve el seno generado en la FPGA medido en banda base. Podemos ver que la frecuencia del seno es de 219.51kHz.

Para medir como funcionan las bandas de trabajo, se configura en el DSP, como ya se ha explicado antes, las diferentes frecuencias de portadora. Para ello, se han ido cambiando los valores de la variable `float TxCarrier`. Primero se prueba en la primera banda (de 262MHz a 435MHz) y luego en la segunda (de 523MHz a 876MHz). (Ver tabla 4.1). Todas las medidas están realizadas en radiofrecuencia.

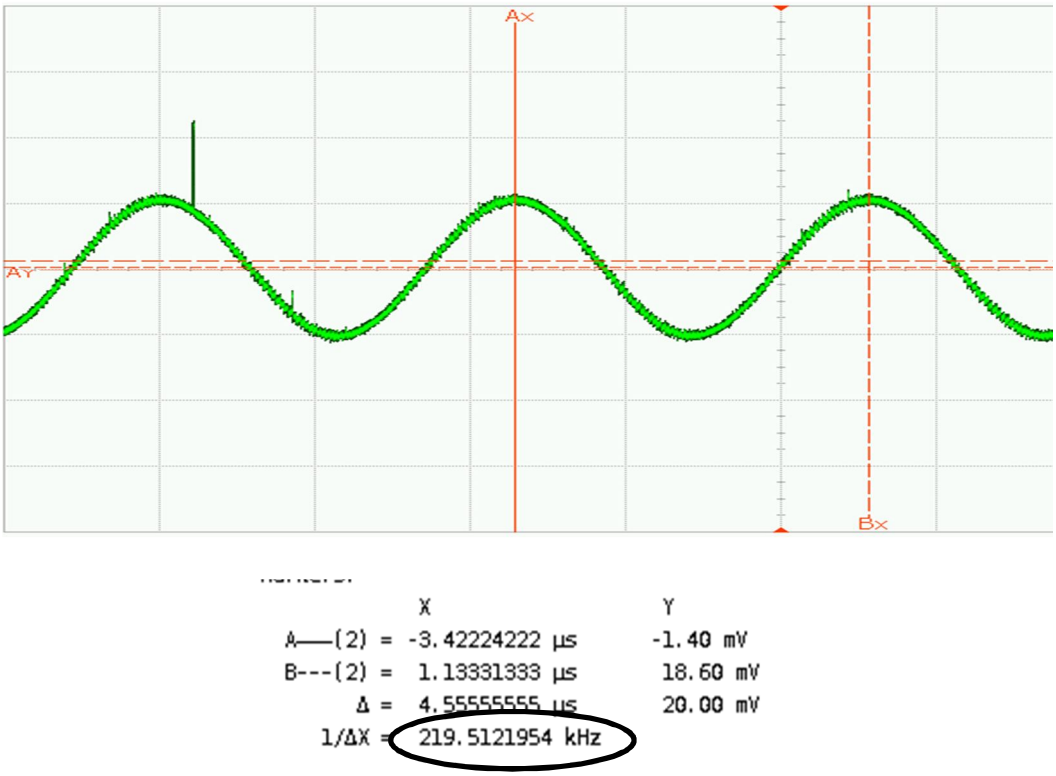

Figura 4.5: Frecuencia del seno enviado en banda base.

Primera Banda		
Prescaler	TxCarrier (MHz)	TxFreq (MHz)
1	200 (por debajo de la banda)	200
1	260 (borde de la banda)	260
1	300	300
1	370	370
1	400	400
1	438 (borde de la banda)	438
1	460 (por encima de la banda)	460
1	510	477
1	600	477
Segunda Banda		
Prescaler	TxCarrier (MHz)	TxFreq (MHz)
0	260 (borde de la banda)	520
0	300	600
0	370	740
0	420	840
0	438 (borde de la banda)	876
0	480 (por encima de la banda)	955
0	500	955

Tabla 4.1: Frecuencias de portadora

Por debajo de la primera banda, a 200MHz, con el `prescaler` activado, la señal modulada (Ver figura 4.6) y portadora (Ver figura 4.7) se observa con inestabilidad. La señal modulada va a la frecuencia del seno generado en la FPGA. La señal portadora va a la frecuencia que se ha programado en el DSP, 200 MHz. En las figuras 4.8 y 4.9, se puede observar la FFT (`Fast Fourier Transform);` los armónicos secundarios son bastante elevados. Si se hace zoom en la figura 4.8, se obtiene la figura 4.9, en donde se puede ver que hay bastantes interferencias, pero la frecuencia de portadora está centrada en 200 MHz.

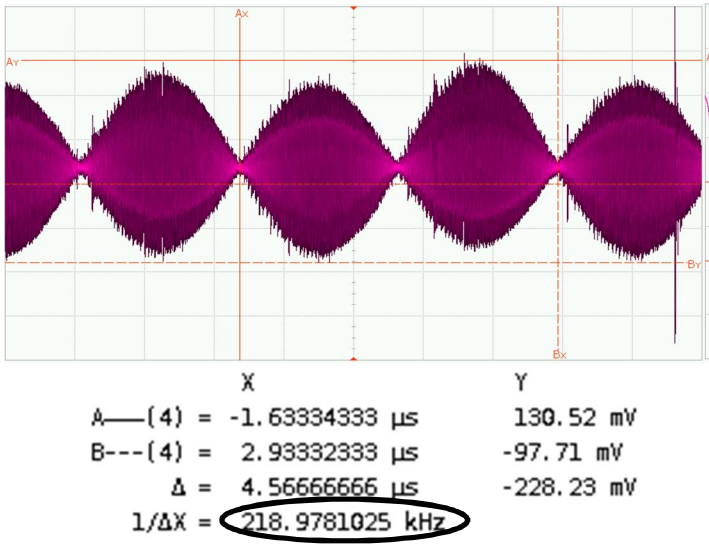

Figura 4.6: Señal modulada a 200 MHz

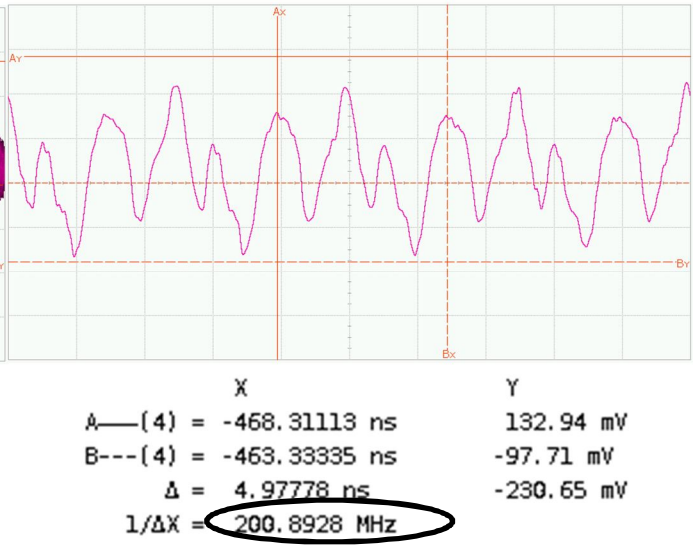

Figura 4.7: Señal portadora a 200 MHz

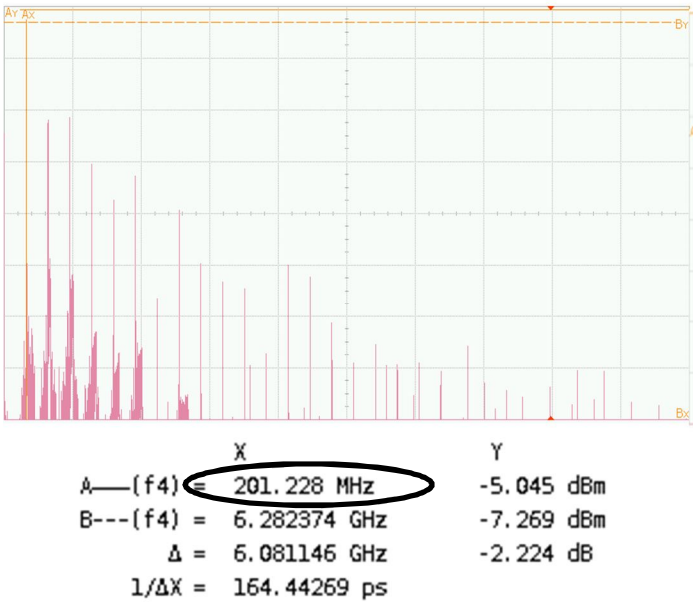

Figura 4.8: FFT a 200 MHz

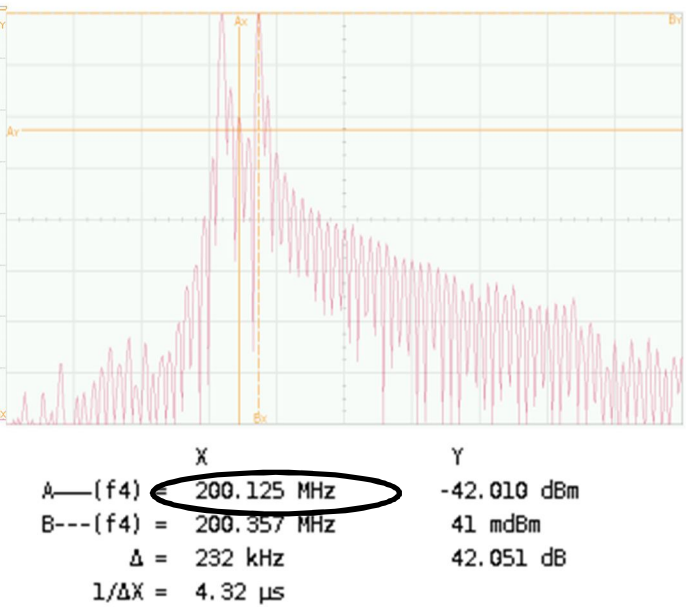

Figura 4.9: FFT zoom a 200 MHz

En el borde de la primera banda, a 260 MHz, con el `prescaler` activado, la señal modulada (Ver figura 4.10) y portadora (Ver figura 4.11) se observa con un poco de inestabilidad pero bastante menos que a 200 MHZ. La señal modulada va a la frecuencia del seno generado en la FPGA. La señal portadora va a la frecuencia que se ha programado en el DSP, 257.14 MHz. En las figuras 4.12 y 4.13, se puede observar el espectro de la señal modulada; los armónicos secundarios son menos elevados. Si se hace zoom en la figura 4.12, se obtiene la figura 4.13, en donde se puede ver claramente la frecuencia de portadora y la frecuencia lateral superior sin ninguna interferencia. Luego el ancho de banda es de 0.44 MHz.

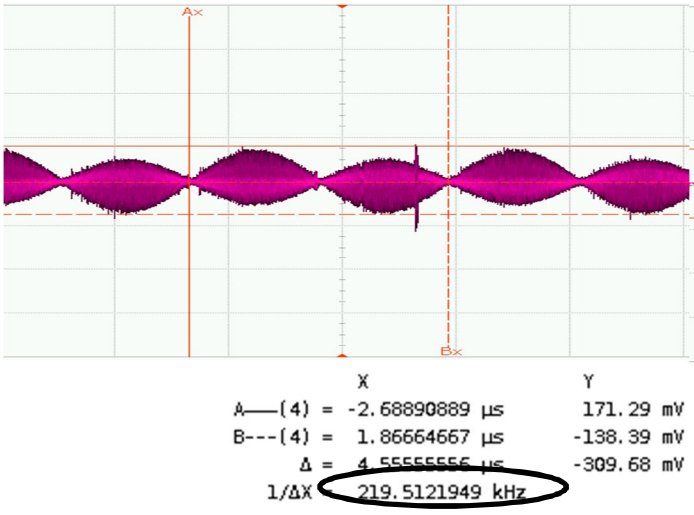

Figura 4.10: Señal modulada 260 MHz

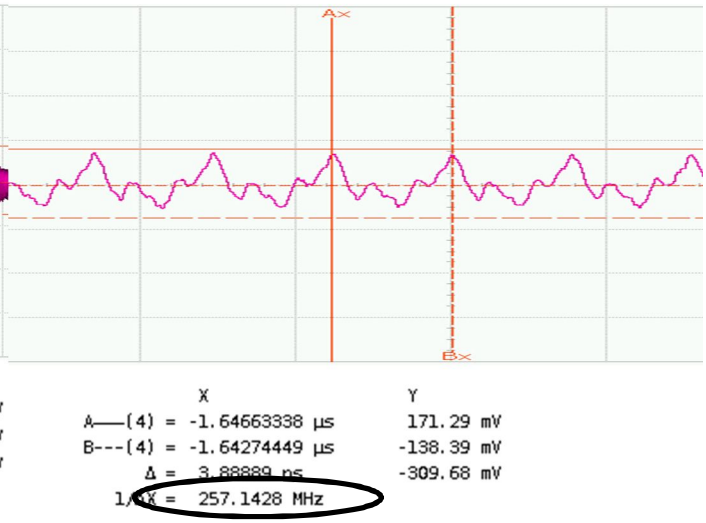

Figura 4.11: Señal portadora 260 MHz

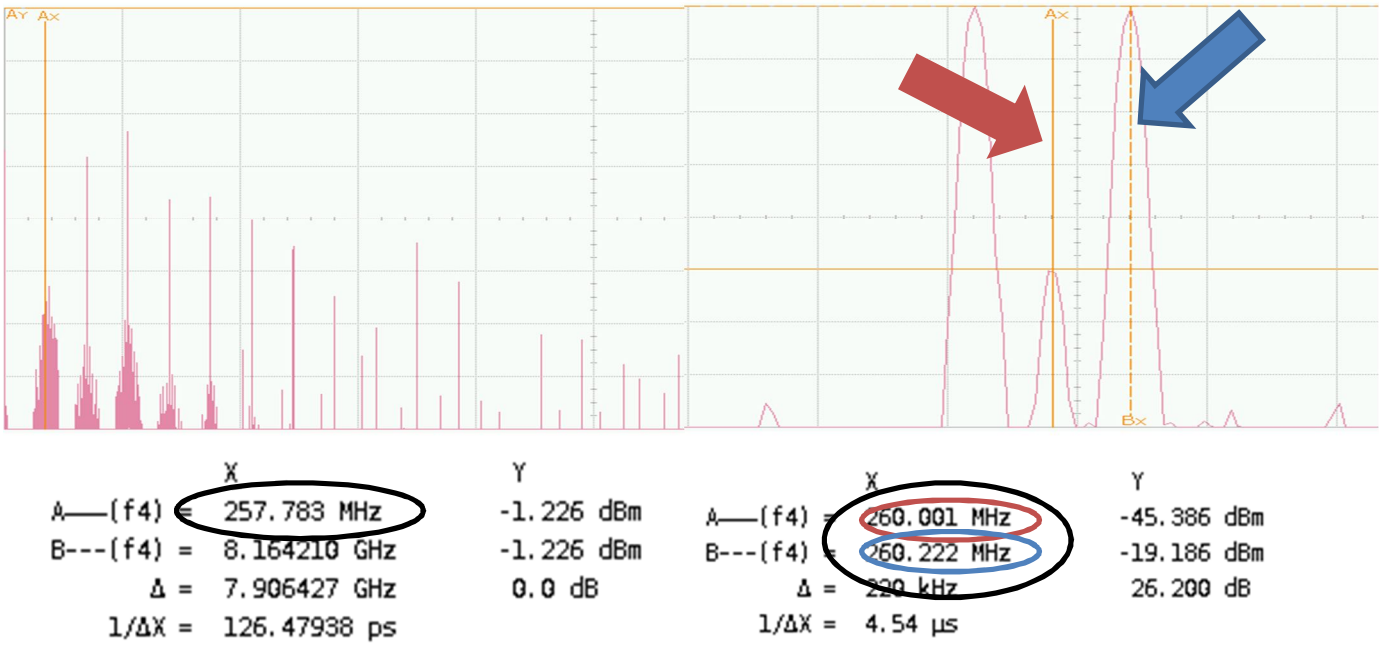

Figura 4.12: FFT a 260 MHz

Figura 4.13: FFT zoom a 260 MHz

En la primera banda en el centro, a 370 MHz, con el `prescaler` activado, la señal modulada (Ver figura 4.14) y portadora (Ver figura 4.15) se observa estable. La señal modulada va a la frecuencia del seno generado en la FPGA. La señal portadora va a la frecuencia que se ha programado en el DSP, 371.901 MHz. En las figuras 4.16 y 4.17, se puede observar el espectro de la señal modulada; hay menos armónicos secundarios. Si se hace zoom en la figura 4.16, se obtiene la figura 4.17, en donde se puede ver claramente la frecuencia de portadora y la frecuencia lateral superior sin ninguna interferencia. El ancho de banda es de 0.446 MHz.

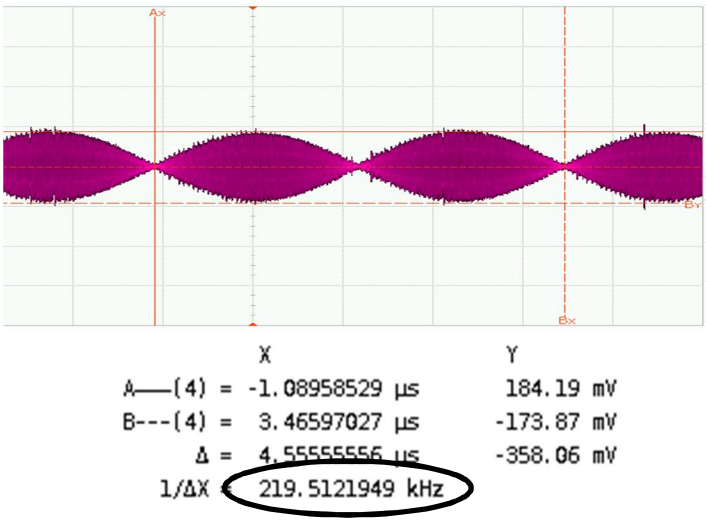

Figura 4.14: Señal modulada 370 MHz

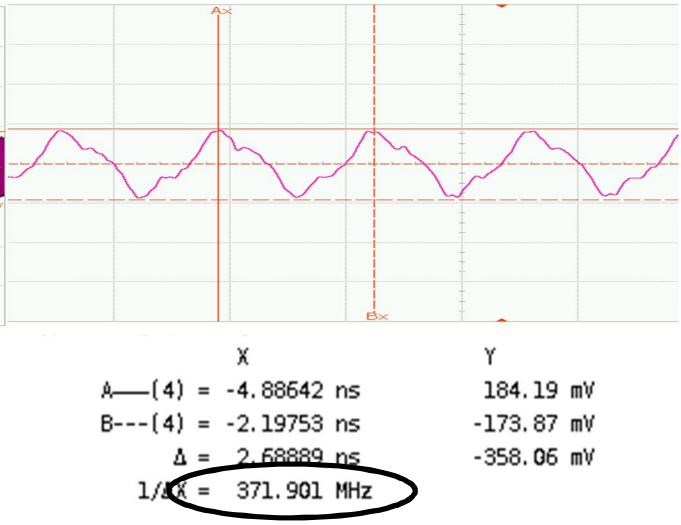

Figura 4.15: Señal portadora 370 MHz

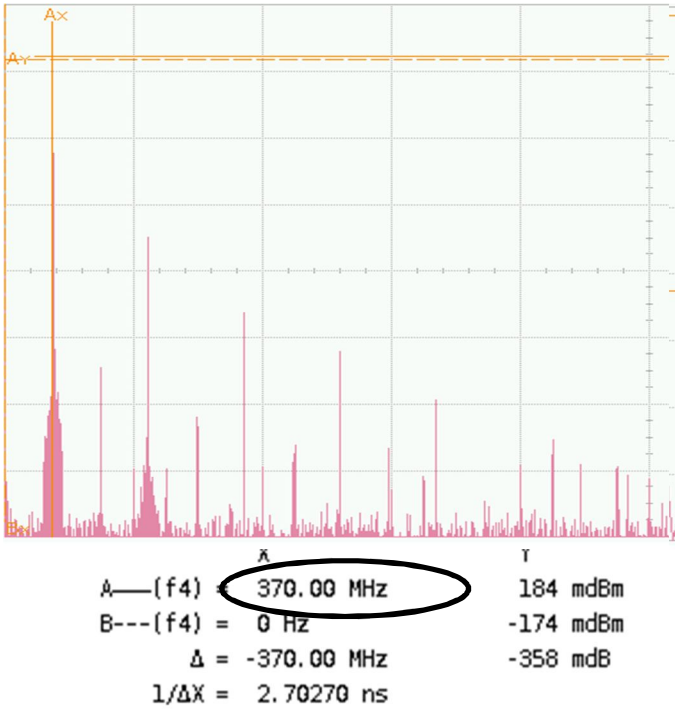

Figura 4.16: FFT a 370 MHz

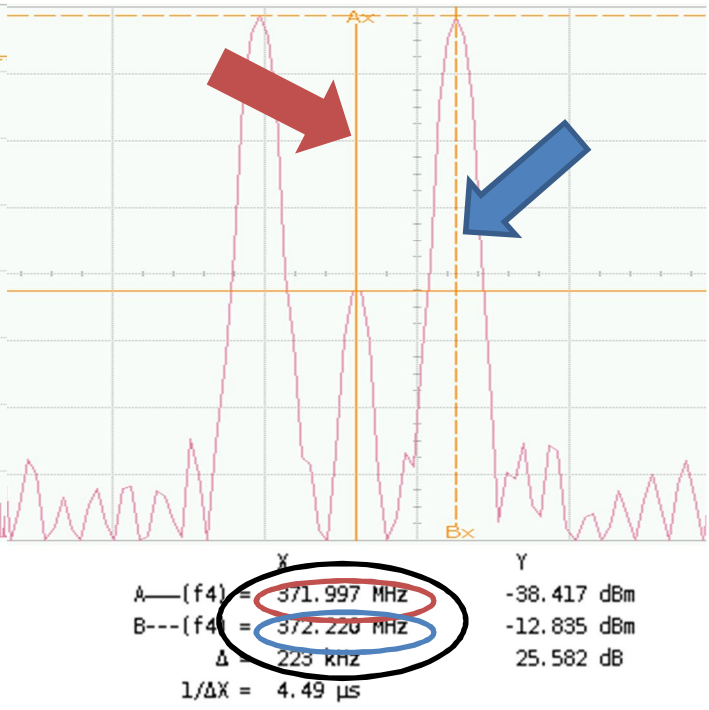

Figura 4.17: FFT zoom a 370 MHz

En el borde superior de la primera banda, a 460 MHZ, con el `prescaler` activado, la señal modulada (Ver figura 4.18) y portadora (Ver figura 4.19) se observa estable. La señal modulada va a la frecuencia del seno generado en la FPGA. La señal portadora va a la frecuencia que se ha programado en el DSP, 479.183 MHz. En las figuras 4.20 y 4.21, se puede observar el espectro de la señal modulada. Si se hace zoom en la figura 4.20, se obtiene la figura 4.21, en donde se ve la frecuencia de portadora y la frecuencia lateral superior, pero hay interferencias.

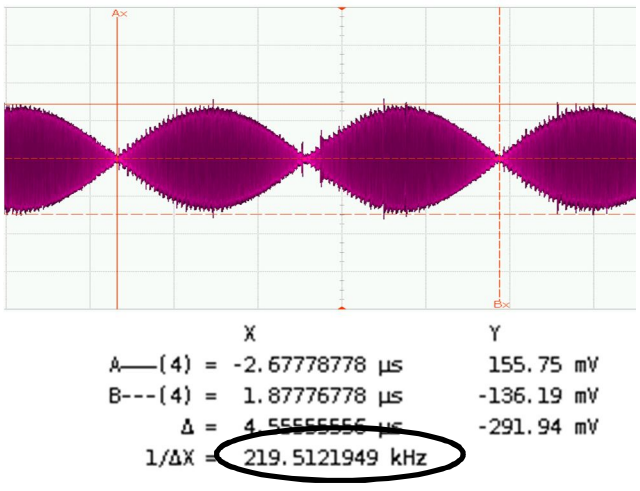

Figura 4.18: Señal modulada 460 MHz

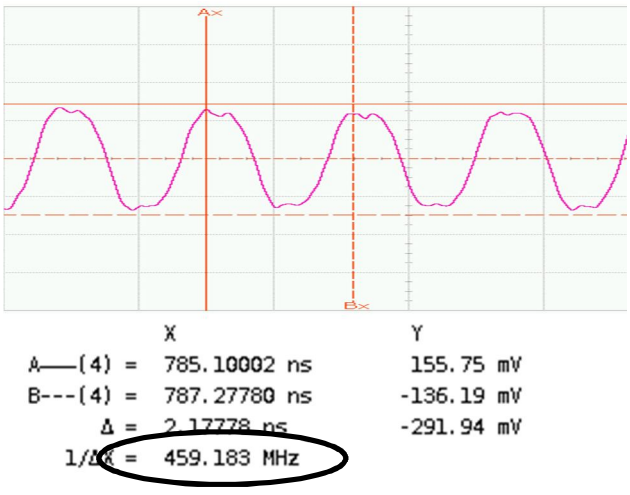

Figura 4.19: Señal portadora 460 MHz

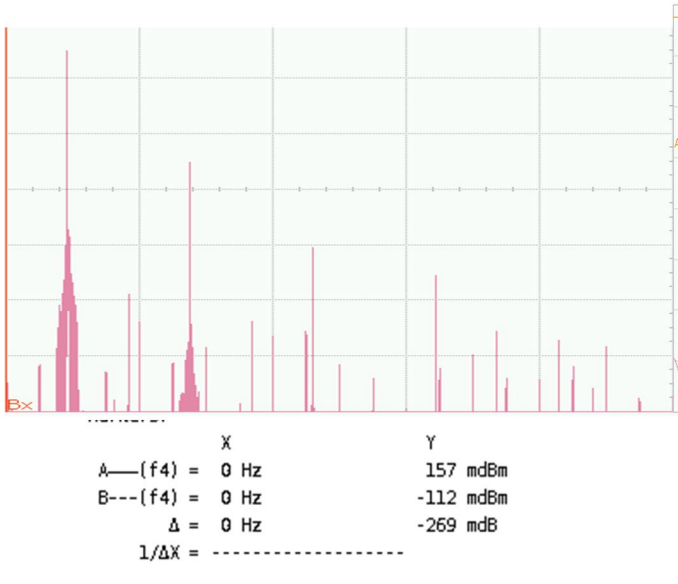

Figura 4.20: FFT a 460 MHz

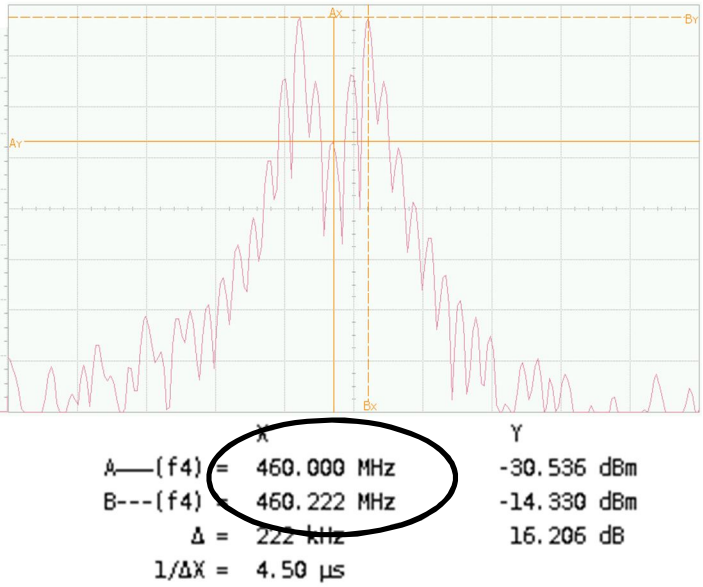

Figura 4.21: FFT zoom a 460 MHz

A 600 MHz, pero con el `prescaler` activado (para usar la primera banda), aunque 600 MHz está en la segunda banda. Se hace esta medida para comprobar los límites de las bandas. En las figuras 4.22 y 4.23, se observan la señal modulada y la portadora. La señal de portadora sale a 478.729 MHz. En las figuras 4.24 y 4.25, se observa el espectro de la señal modulada. Se puede ver que hay bastante interferencia. A partir de 460 MHz, con el `prescaler` activado, la frecuencia de portadora siempre saldrá alrededor de 478 MHz, aunque en el DSP suba la frecuencia.

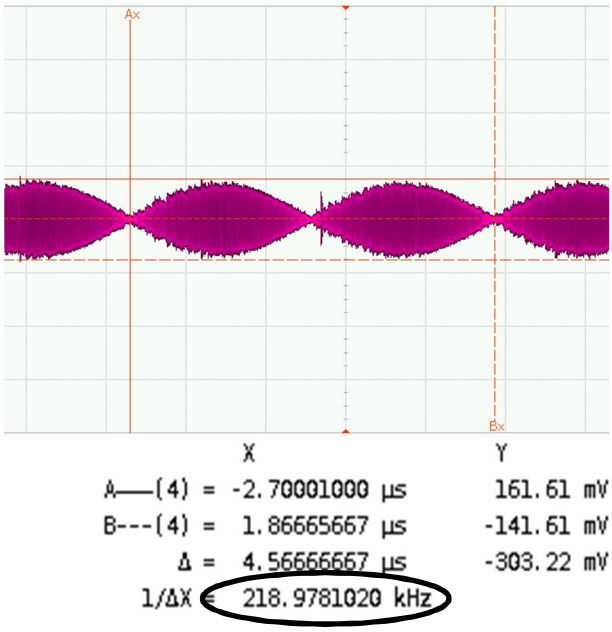

Figura 4.22: Señal modulada 600 MHz

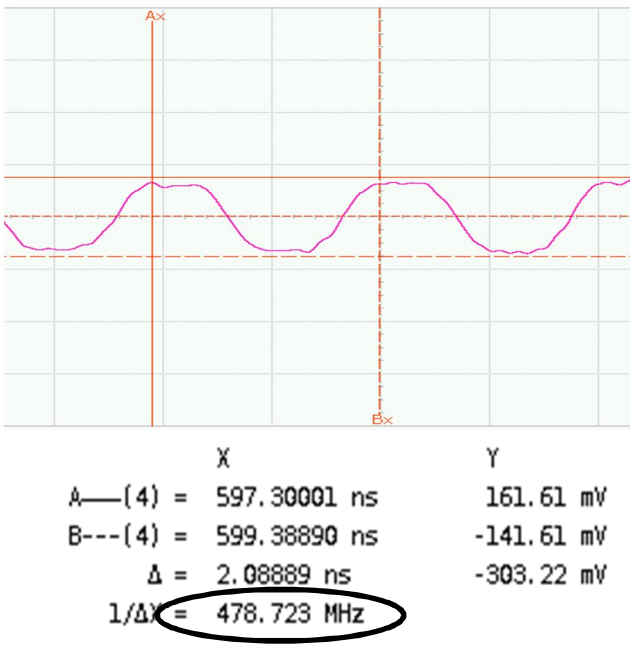

Figura 4.23: Señal portadora 600 MHz

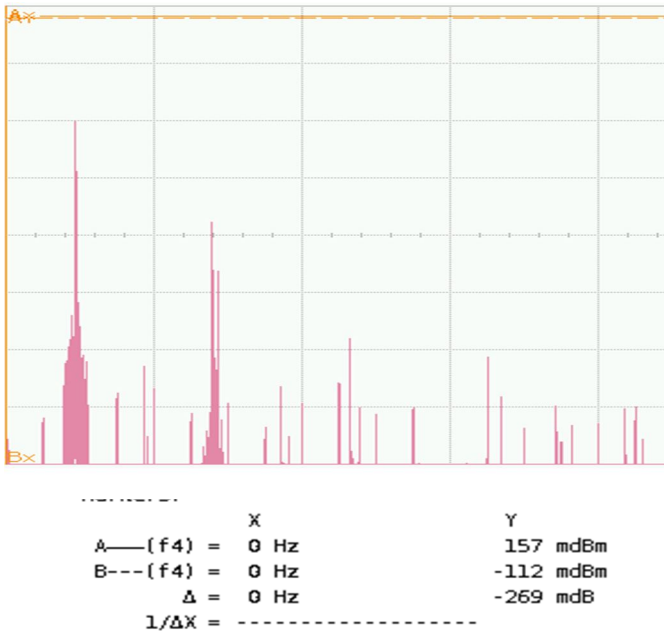

Figura 4.24: FFT a 600 MHz

Figura 4.25: FFT zoom a 600 MHz

En el borde inferior de la segunda banda, a 520 MHz, con el `prescaler` desactivado. La señal modulada (Ver figura 4.26) va a la frecuencia del seno generado en la FPGA. La señal portadora (Ver figura 4.27) va a la frecuencia que se ha programado en el DSP, 523.253 MHz. En las figuras 4.28 y 4.29, se puede observar la FFT. Si se hace zoom en la figura 4.28, se obtiene la figura 4.29, en donde se puede ver la frecuencia de portadora que está centrada en 520 MHz y la frecuencia lateral superior a 520.224 MHz.

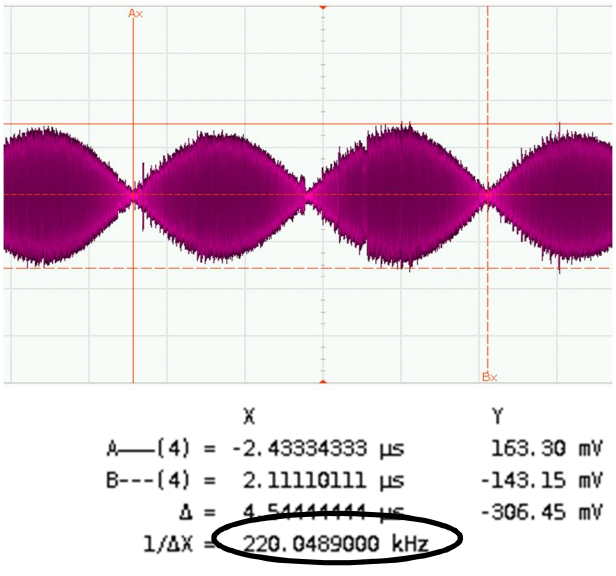

Figura 4.26: Señal modulada 520 MHz

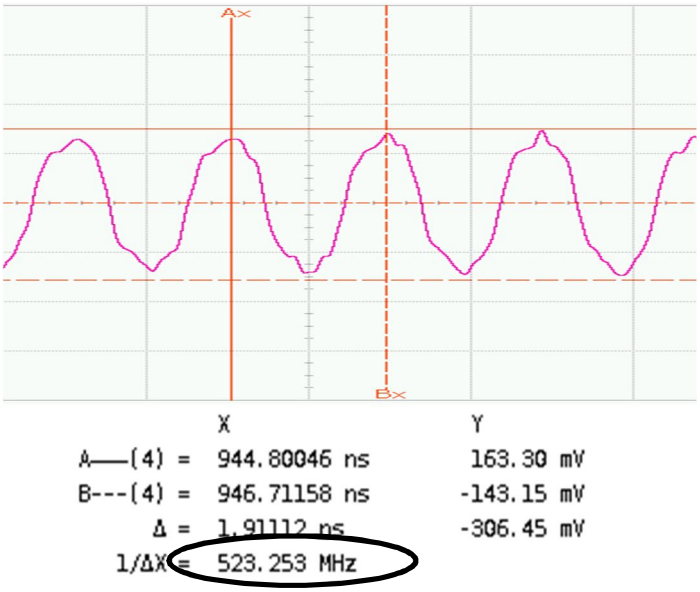

Figura 4.27: Señal portadora 520 MHz

Figura 4.28: FFT a 520 MHz

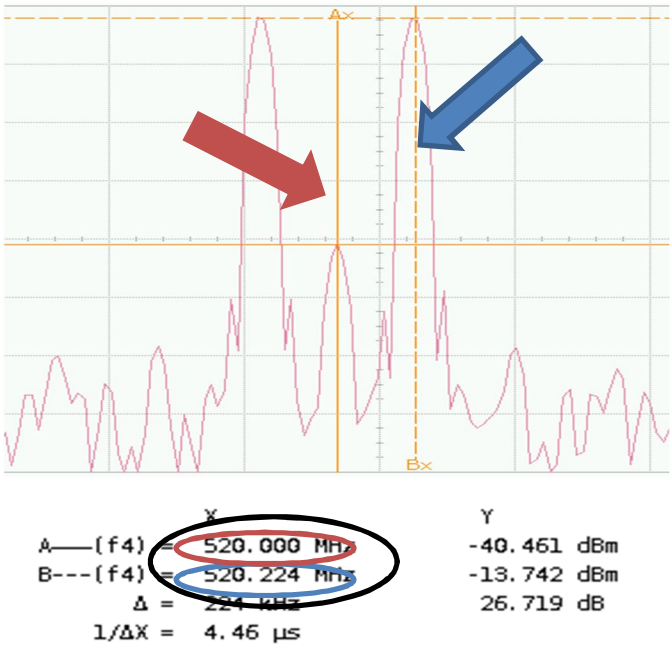

Figura 4.29: FFT zoom a 520 MHz

En la mitad de la segunda banda, a 740 MHz, con el `prescaler` desactivado, la señal modulada (Ver figura 4.30) y portadora (Ver figura 4.31) se observa estable. La señal modulada va a la frecuencia del seno generado en la FPGA. La señal portadora va a la frecuencia que se ha programado en el DSP, 743.804 MHz. En las figuras 4.32 y 4.33, se puede observar el espectro de la señal modulada. Si se hace zoom en la figura 4.32, se obtiene la figura 4.33, en donde se puede ver claramente la frecuencia de portadora 740 MHzy la frecuencia lateral superior 740.220 MHz. El ancho de banda es de 0.44 MHz.

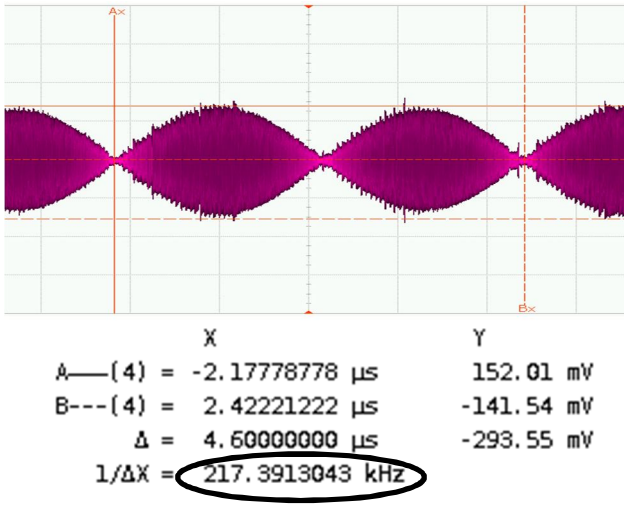

Figura 4.30: Señal modulada 740 MHz

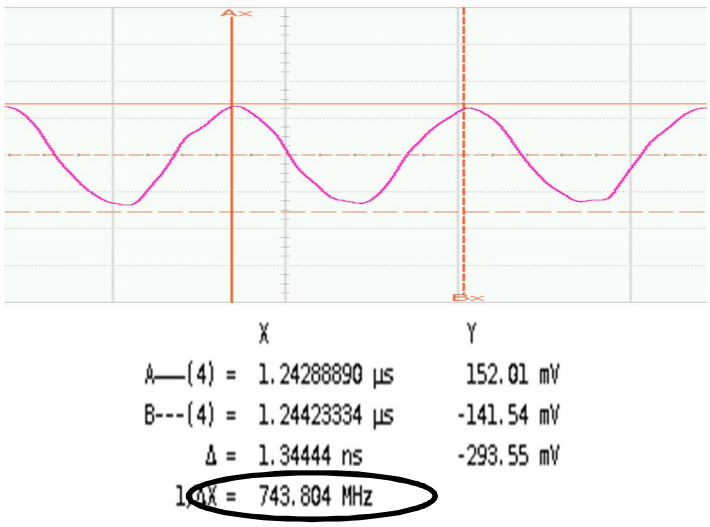

Figura 4.31: Señal portadora 740 MHz

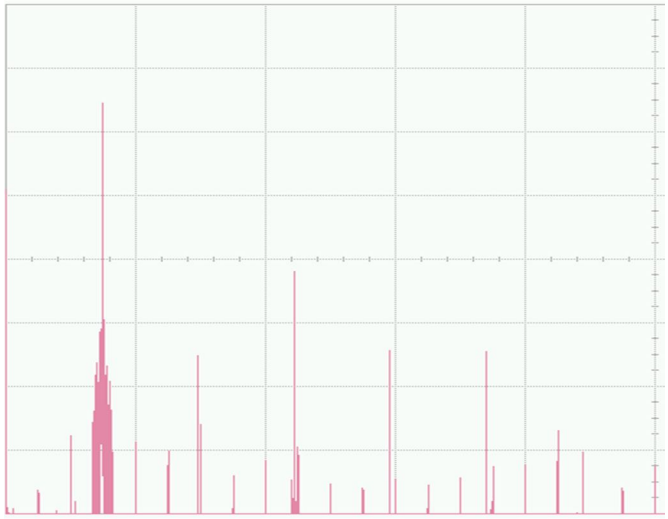

Figura 4.32: FFT a 740 MHz

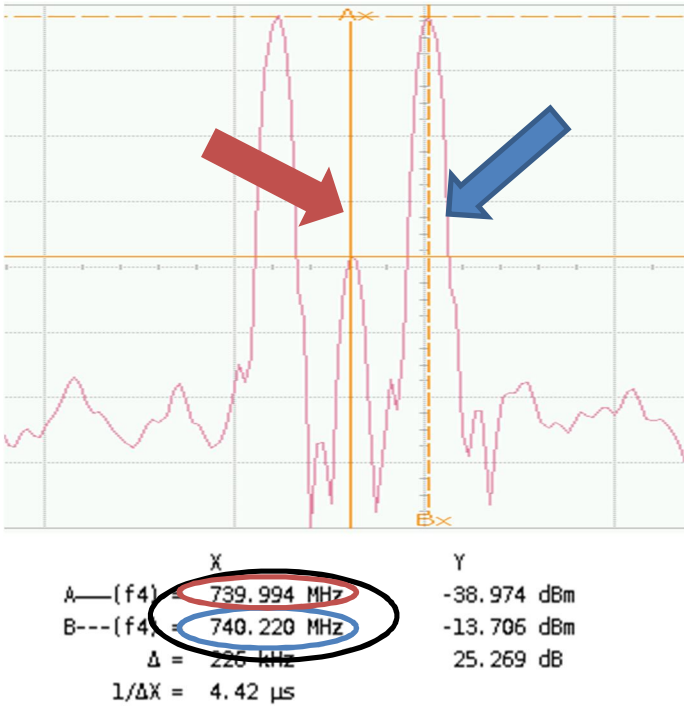

Figura 4.33: FFT zoom a 740 MHz

En el borde superior de la segunda banda, a 876 MHz, con el `prescaler` desactivado. La señal modulada (Ver figura 4.34) va a la frecuencia del seno generado en la FPGA. La señal portadora (Ver figura 4.35) va a la frecuencia que se ha programado en el DSP. En las figuras 4.36 y 4.37, se puede observar el espectro de la señal modulada. En ellas se ve la frecuencia de portadora, 876.003 MHz y la frecuencia lateral superior, 876.223 MHz.

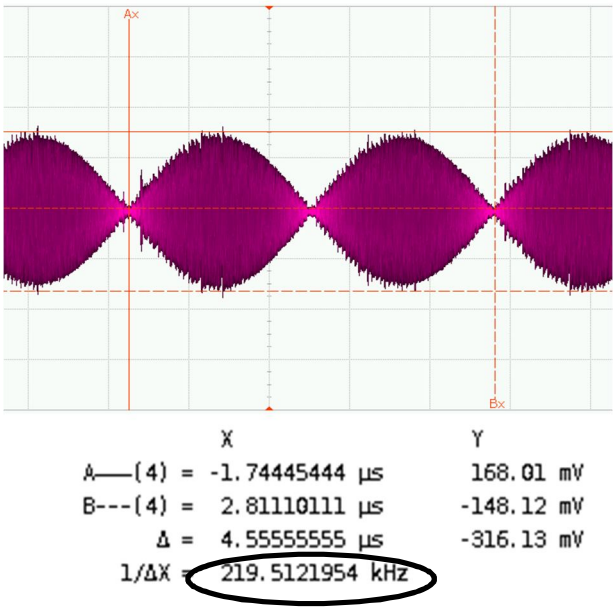

Figura 4.34: Señal modulada 876 MHz

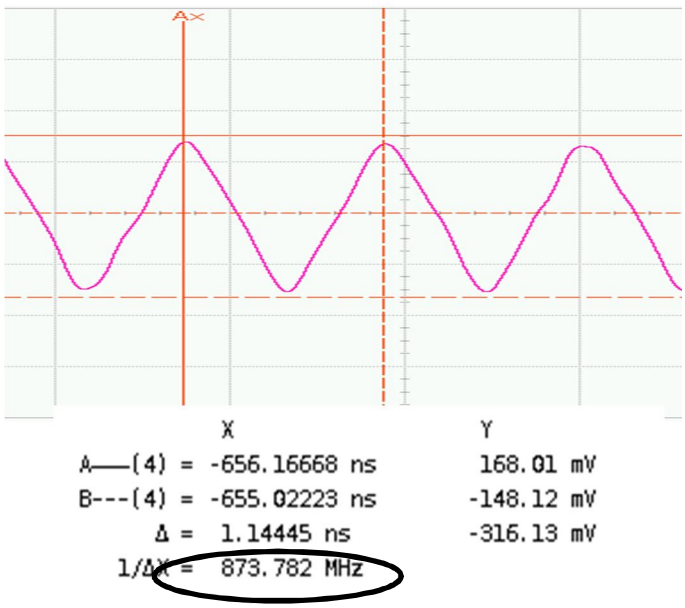

Figura 4.35: Señal portadora 876 MHz

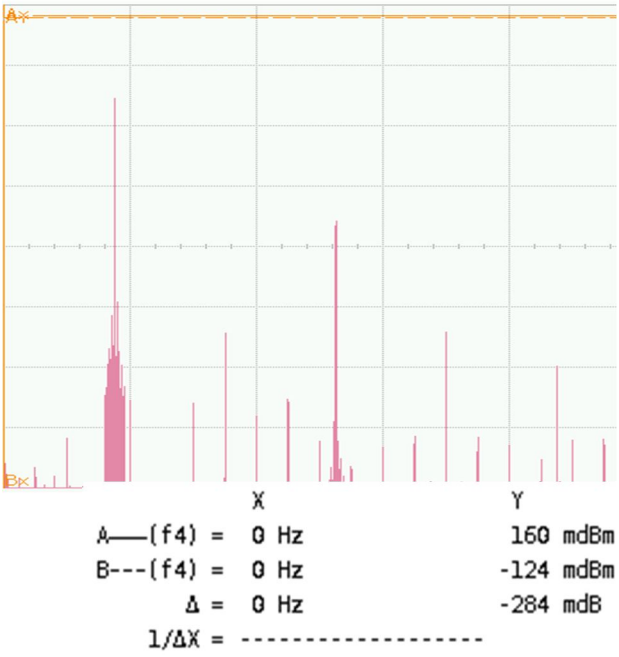

Figura 4.36: FFT a 876 MHz

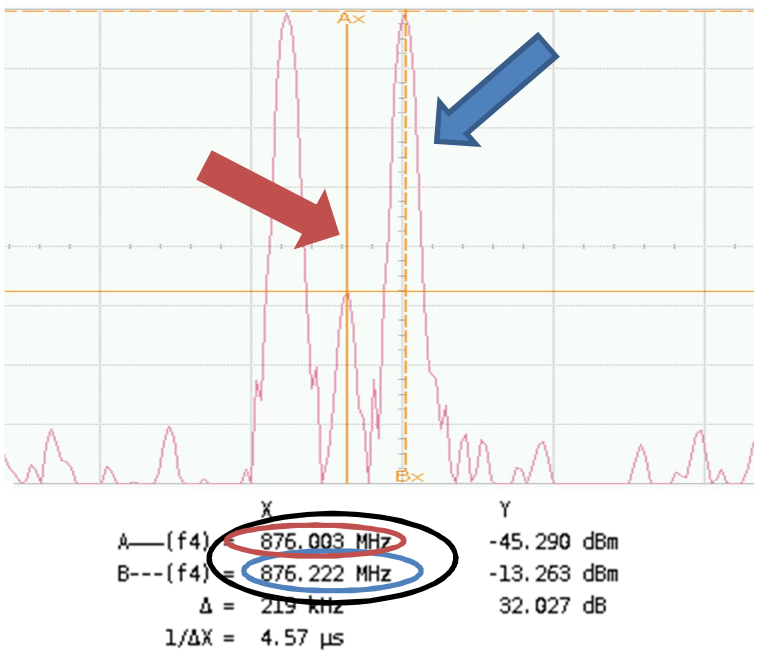

Figura 4.37: FFT zoom a 876 MHz

Por encima de la segunda banda, a 960 MHz, la señal modulada (Ver figura 4.38) va a la frecuencia del seno generado en la FPGA. La señal portadora (Ver figura 4.39) va a 957.451 MHz. Aunque la frecuencia de portadora se programe más alta, siempre va a salir alrededor de 955 MHz. En las figuras 4.40 y 4.41, se puede observar el espectro de la señal modulada. En ellas se ve la frecuencia de portadora. Además la señal es bastante inestable.

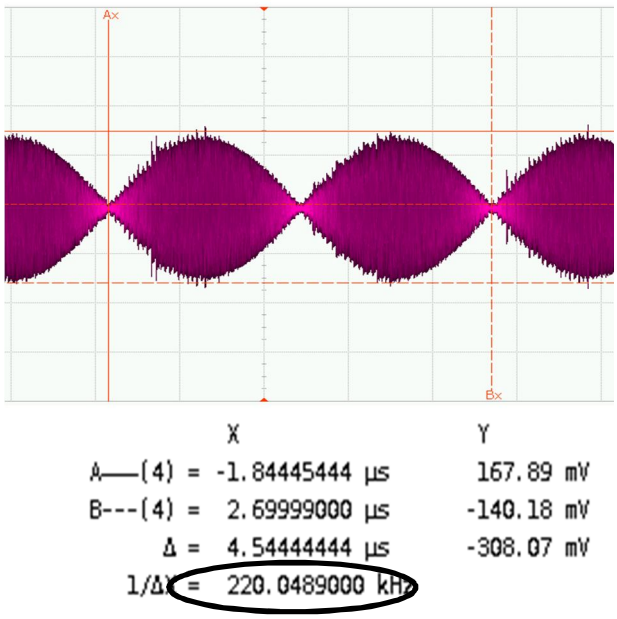

Figura 4.38: Señal modulada 960 MHz

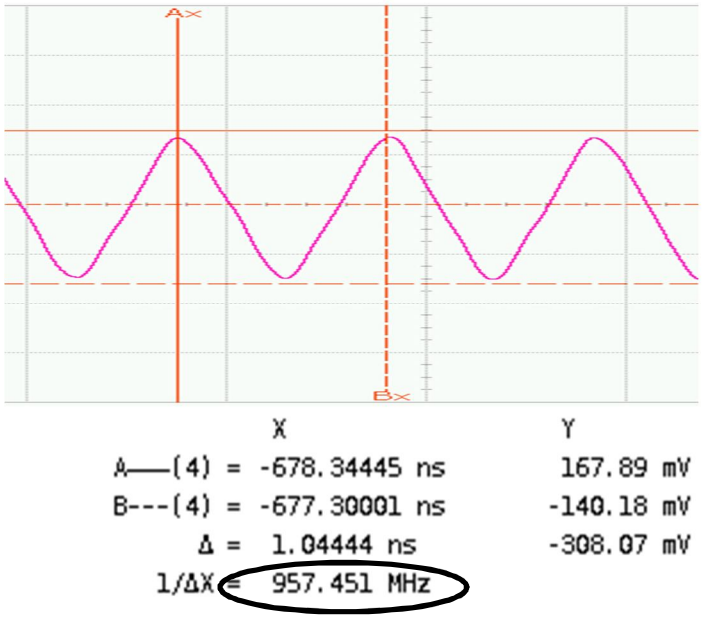

Figura 4.39: Señal portadora 960 MHz

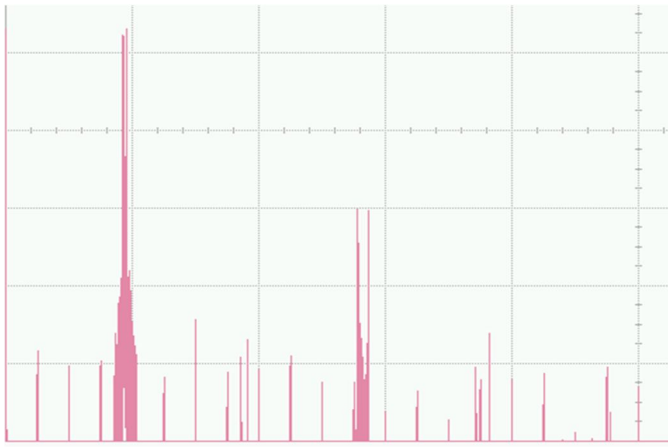

Figura 4.40: FFT a 960 MHz

Figura 4.41: FFT zoom a 960 MHz

Como se puede apreciar en las imágenes, las bandas de trabajo son un poco más grandes de lo que pone en las especificaciones.

En frecuencias comprendidas en cada banda, el primer armónico siempre sale a la frecuencia de portadora y sin inestabilidad.

En los bordes de las bandas, se puede ver que el primer armónico está situado a la frecuencia de portadora. Pero cuando se sale por encima de cada banda, el primer armónico se sitúa a una frecuencia determinada y aunque se aumente no va a ser sobrepasada. En la primera banda, con el `prescaler` activado, el primer armónico siempre saldrá como máximo a 477MHz. En la segunda banda, con el `prescaler` desactivado, la máxima frecuencia de portadora será 955MHz. Además se puede apreciar la existencia de inestabilidad.

Sin embargo, cuando se sale por debajo de cada banda, el primer armónico sale a la frecuencia de la señal de portadora que se ha programado en el DSP, aunque se observa inestabilidad.

Por lo tanto, las bandas de trabajo empeoran más por arriba que por abajo.

Valores de ganancia del DAC

Como se ha comentado anteriormente, cambiando los valores de los parámetros `Fine` y `Coarse` en `conv_mod_SetDACChanGain(param_cnt, channel, gain_type, gain);` se puede variar la ganancia. (Ver tabla 4.2).

Para medir estos valores de ganancia, se ha elegido la primera banda de trabajo a 370 MHZ.

	Canal A		Canal B		
Opción	Fine	Coarse	Fine	Coarse	
1	0	14	0	14	
2	0	15	0	15	
3	127	15	127	15	Max. Ganancia
4	0	0	0	0	Min. Ganancia
5	0	2	0	2	
6	0	10	0	4	

Tabla 4.2: Valores ganancia del DAC

En primer lugar, se muestra la salida de la segunda opción ya que es muy semejante de la primera y es innecesario mostrarlo dos veces. En la figura 4.42, se puede observar el seno medido en banda base, sale a 353 mVpp. En la figura 4.43, aparece la señal modulada a 393 mVpp. El espectro de la señal modulada se puede ver en la figura 4.44, hay una diferencia de 12.74 dB entre el primer armónico y el segundo. El primer armónico sale a -12.58 dBm. La diferencia entre la primera opción y la segunda es de 35 mVpp en la señal modulada.

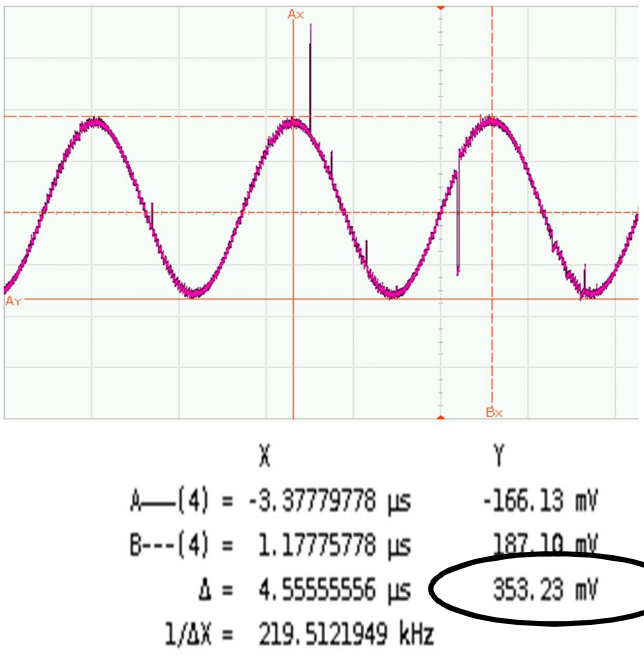

Figura 4.42: Señal en banda base 2ºop

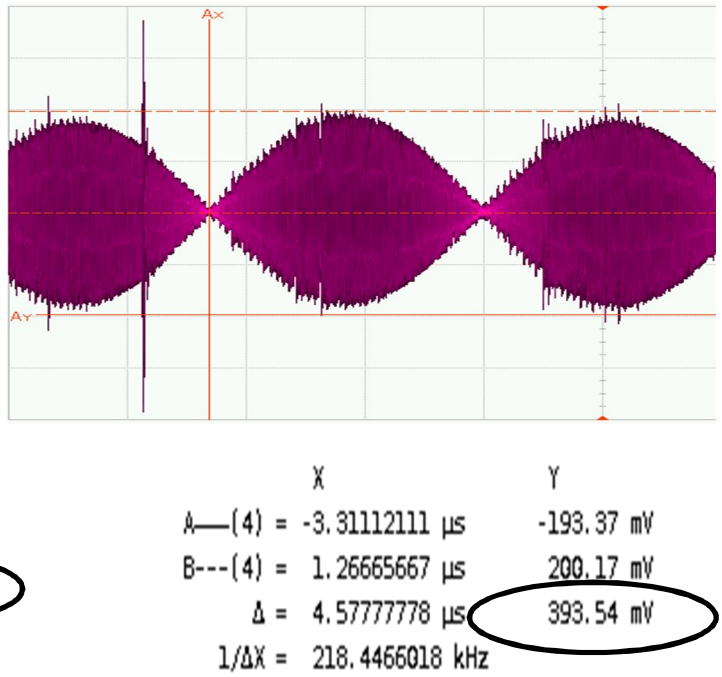

Figura 4.43: Señal modulada 2ºop

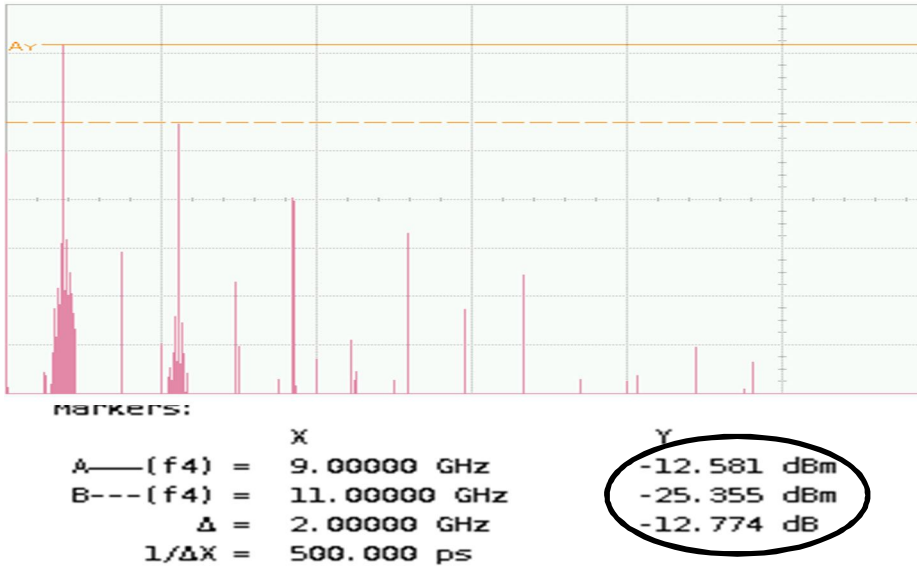

Figura 4.44: FFT 2ºop.

A continuación se muestra la opción de máxima ganancia. En la figura 4.45, se observa el seno en banda base, sale a 377.42 mVpp. En la figura 4.46, aparece la señal modulada a 416.13 mVpp, no satura. En la figura 4.47 se puede ver el espectro de la señal modulada, hay una diferencia de 15.419 dB entre el primer armónico y el segundo. El primer armónico sale a -11.9 dBm.

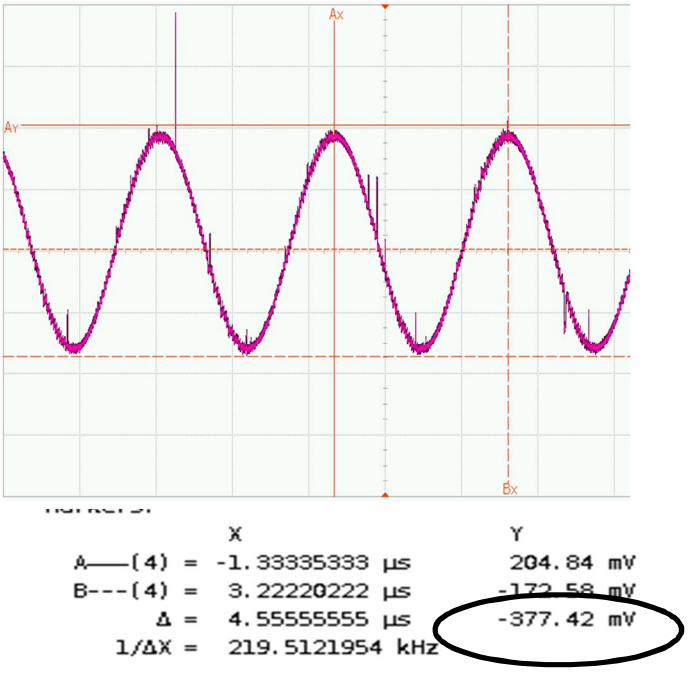

Figura 4.45: Señal en banda base 3ºop

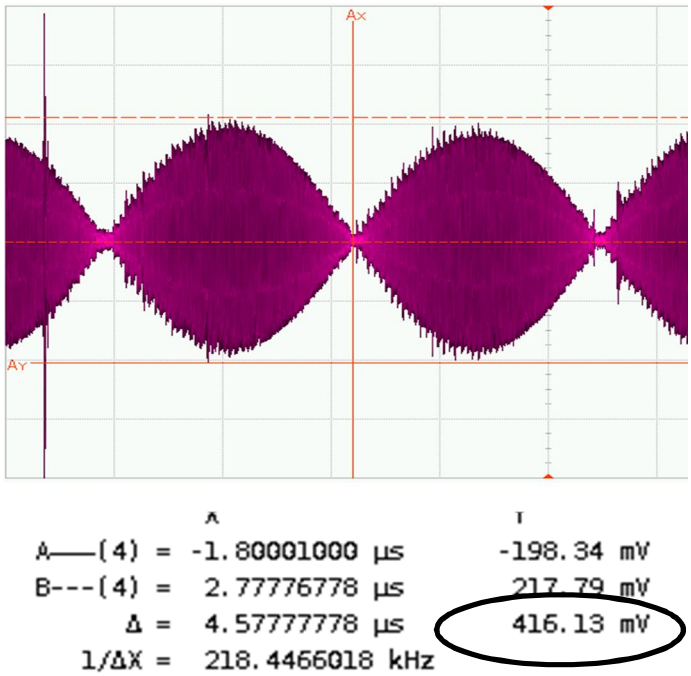

Figura 4.46: Señal modulada 3ºop

Figura 4.47: FFT 3ºop.

La opción de mínima ganancia se muestra en las figuras 4.48, 4.49, 4.50. El seno en banda base sale a 25.9 mVpp. La señal modulada a 30.9 mVpp. En el espectro de la señal modulada, hay una diferencia de 10 dB entre el primer y el segundo armónico. El primer armónico sale a -36 dBm.

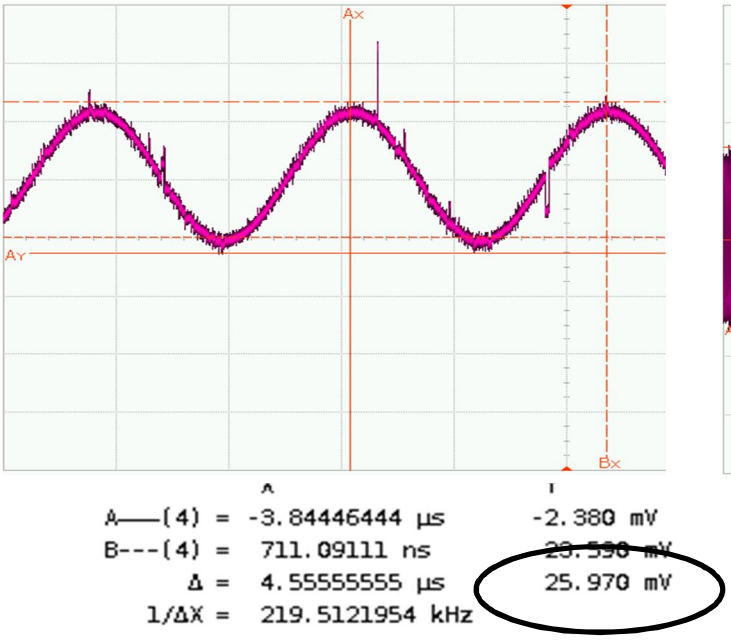

Figura 4.48: Señal en banda base 4ºop

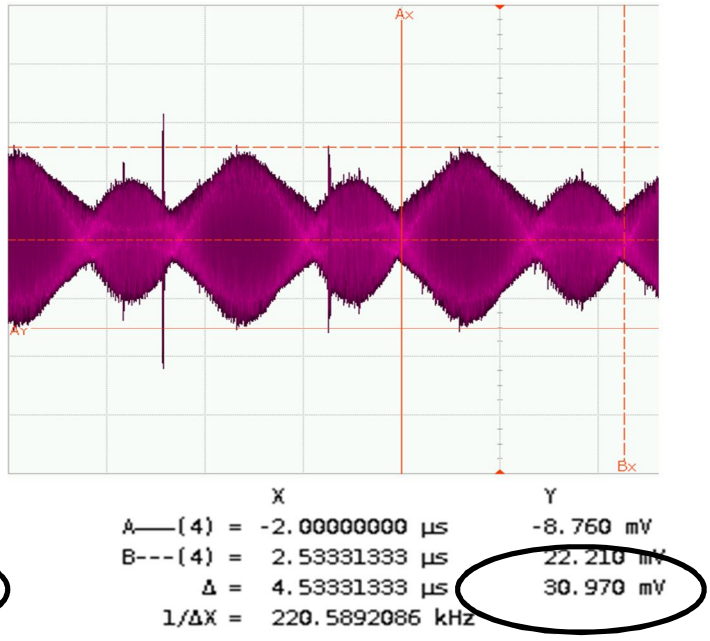

Figura 4.49: Señal modulada 4ºop

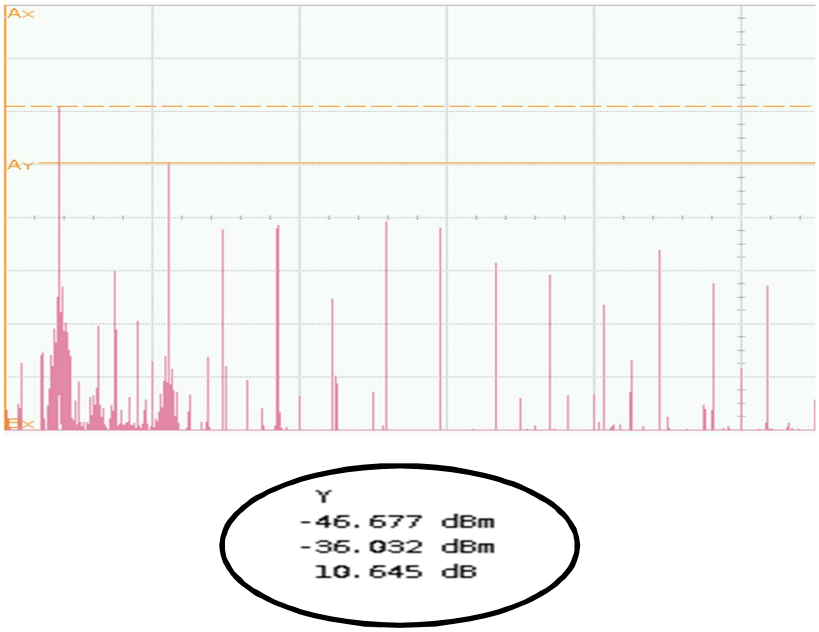

Figura 4.50: FFT 4ºop.

La quinta opción se muestra en la figura 4.51. Se observa el seno en banda base, sale a 70.640 mVpp. En la figura 4.52, aparece la señal modulada a 78.39 mVpp. En la figura 4.53 se puede ver el espectro de la señal modulada, hay una diferencia de 14 dB entre el primer armónico y el segundo. El primer armónico sale a -26dBm.

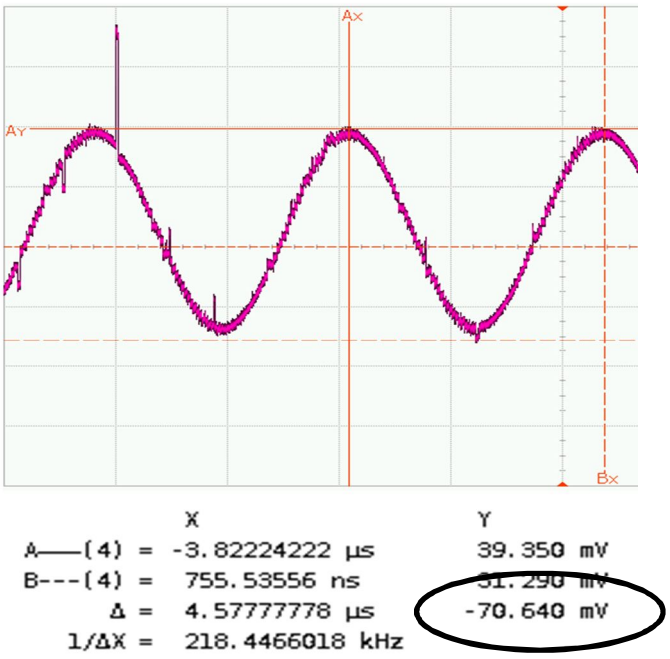

Figura 4.51: Señal en banda base 5ºop

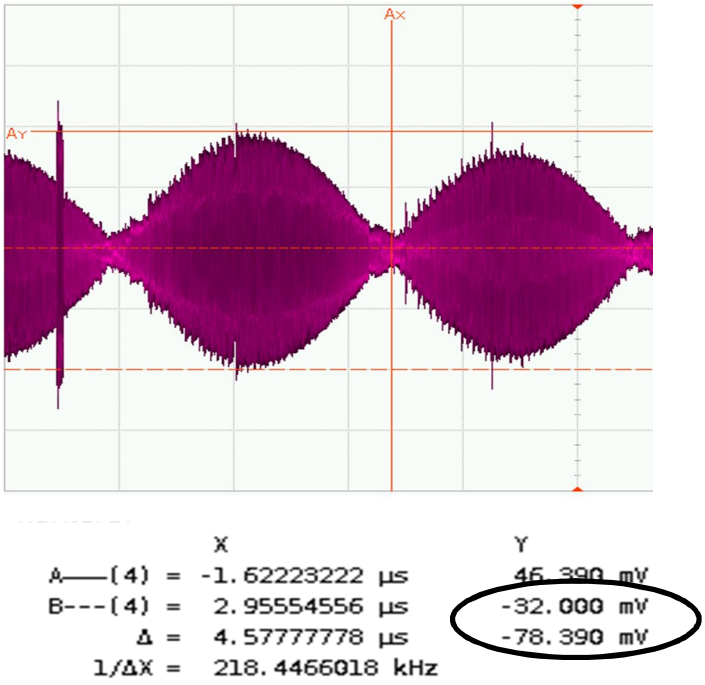

Figura 4.52: Señal modulada 5ºop

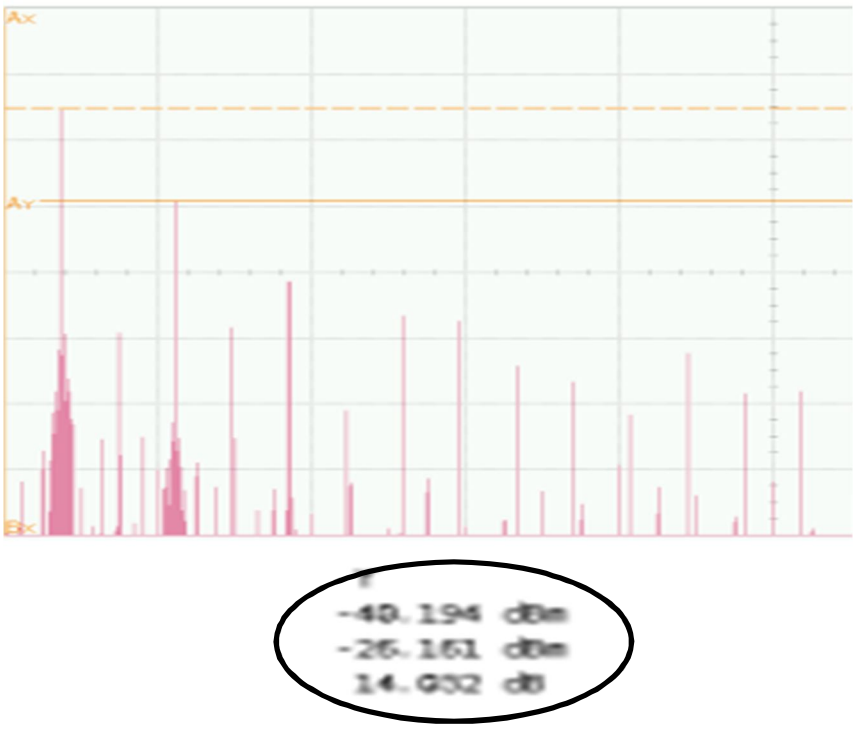

Figura 4.53: FFT 5ºop.

Examinando las imágenes, se puede observar que al incrementar el ajuste grueso (`Coarse`) en un punto, se consigue un aumento de 35 mVpp. Lo que significa que cada vez que se aumente un punto en el ajuste grueso la señal se va a amplificar un 8.9%.

Si además se aumenta el ajuste fino (`Fine`) al máximo se consigue un aumento de 23 mVpp, lo que implica un 5.5% más de amplitud.

La diferencia entre el primer y segundo armónico, independientemente de los ajustes, tiene un valor entre 10dBm y 14 dBm.

Cuando el ajuste se hace para que funcione a máxima ganancia, no satura.

5 CONCLUSIONES

En este trabajo se ha intentado dar una idea de cómo utilizar el dispositivo Lyrtech SFF SDR. Desde la instalación de todos los programas que se utilizan en su configuración hasta el funcionamiento de las bandas de transmisión entre otras cosas. Se ha programado un seno en la FPGA, se ha configurado todos los módulos del dispositivo con el DSP. Además, se ha hecho un pequeño estudio de las tecnologías basadas en DSP y FPGA, haciendo hincapié en los principales fabricantes.

La mayor parte de la dificultad de este trabajo se encuentra en la familiarización con el dispositivo. La información es muy escasa, basándose únicamente en los manuales que ofrece el fabricante, en los que no hay apenas información técnica. La instalación de los programas que se utilizan fue bastante complicada, no había licencias, faltaban librerías,… Además de todo esto, la configuración del seno en la FPGA, fue muy complicada ya que no tenía ningún conocimiento sobre VHDL. Debido a estas dificultades, gran parte de este trabajo esta basado en manuales de cómo instalar los programas, cómo cargar los ejecutables en la placa, cómo se generan IP Core, para que en un futuro no se pierda tanto tiempo en familiarizarse con el dispositivo ya que no hay ningún tipo de información en Internet o en foros.

Los resultados obtenidos concuerdan con los manuales de usuario, incluso con mejores características.

Para finalizar, se sugiere la adquisición de otro dispositivo para poder hacer trabajos con los dos juntos y hacer comparaciones uno con el otro.

6 FASES DEL TRABAJO

El desarrollo de este trabajo se ha desarrollado entre Febrero 2012 y Junio 2012. Durante este periodo, el trabajo ha ido evolucionando en diferentes fases.

Primera fase: Familiarización con el dispositivo

La primera fase y la más larga, ya que ha ocupado la mitad del tiempo de duración de este trabajo (dos meses y medio), se ha dedicado al aprendizaje del dispositivo.

Se inició con la lectura de los manuales ofrecidos por el fabricante. Seguidamente se comenzó con las herramientas de desarrollo. El principal problema fue que estas herramientas sólo se encontraban disponibles en el laboratorio, por lo que se decidió instalarlas en un ordenador portátil para poder disponer de ellas en cualquier lugar. Hubo que descargar los programas con las licencias, lo que llevó alrededor de 20 días hasta que estuvo todo correctamente instalado.

En esta fase, se intentó probar los ejemplos suministrados por el fabricante en los manuales, pero ninguno de ellos funcionó. Hubo que ponerse en contacto con Lyrtech pero no fue de gran ayuda.

Finalmente, se recurrió a un proyecto realizado anteriormente [9] y se contactó con su autor que muy amablemente ayudó con las numerosas dudas.

Segunda fase: Implementación de una señal en el dispositivo

Esta segunda fase duró un mes. Cuando se empezó, se deseaba generar un seno directamente en el DSP, pero fue imposible, daba numerosos errores de librerías que no se pudieron resolver. Entonces se pensó en generar la señal en la FPGA.

El problema de generar la señal en la FPGA fue que no se tenía ningún conocimiento previo del lenguaje de descripción de VHDL. Supuso un gran esfuerzo, pero con la ayuda de proyectos anteriores y de manuales de VHDL como [13] se consiguió mandar el seno.

Cuando ya se tenía el seno generado, se configuraron todos los módulos del DSP.

Tercera fase: Prueba del dispositivo

Esta fase fue la más fácil. Una vez generado todos los archivos ejecutables, se cargaron en el dispositivo y simplemente había que medir. Esta fase se extendió 20 días.

Cuarta fase: Escritura de la memoria del trabajo

La última fase ha llevado un mes. Había que poner en orden todo lo hecho durante los meses anteriores y explicarlo de forma clara y sencilla, intentando que en un futuro pueda ser útil para trabajos con este dispositivo.

7 APÉNDICES

Apéndice 1: Presupuesto

En este apartado se detalla el presupuesto relativo a la realización del presente trabajo fin de grado. Se contabiliza el coste personal del desarrollador y el material empleado.

A los costes directos de personal y material, se suman los costes indirectos producidos durante el periodo de trabajo. Estos costes los estimaremos en torno al 20% de la suma de los costes personal y material.

Costes de material

La realización de este TFG ha contado con recursos materiales. Estos costes tendrán en cuenta su amortización en el tiempo. La fórmula empleada para el cálculo de la amortización es la siguiente:

$\frac{A}{B} \times C$ A = nº de meses desde la fecha de facturación en el equipo es utilizado

B = periodo de depreciación (60 meses)

C = coste del equipo (sin IVA)

Para el cálculo de la amortización se considera una vida útil de todo el material de 5 años (60 meses). El IVA (18%) se incluye en costes indirectos.

En la siguiente tabla (ver tabla A1.1) se recogen los costes de material desglosados en equipo informático, herramientas de desarrollo, laboratorio, documentación y otros gastos (impresión de documentos, desplazamientos, etc.). Alcanzan un total de 51879,65€.

	Coste	Duración TFG	Periodo de depreciación	Coste imputable
Ordenador	600€	4	60	40€
Dispositivo SFF SDR y licencias	8.372,20€	4	60	558,15€
Osciloscopio Infinium DSO90604A	4.1967,45€	4	60	2.797,83€
Laboratorio	160€/mes	4	-	640€
Documentación	100€	-	-	100€
Otros gastos	200€	-	-	200€
			Total	4.335,98€

Tabla A1.1: Costes de material

Costes de personal

El coste del personal está principalmente asociado al trabajo realizado por un Ingeniero Junior y un Jefe de Proyecto (el tutor). Teniendo en cuenta que el coste por hora de un Ingeniero Junior se estima en 25€ brutos y la de un Jefe

de Proyecto en 40€ brutos, y que aproximadamente en 4 meses se han dedicado 20 horas a la semana.

Con todo esto el coste de personal asciende a 20.800€.

Costes totales

Sumando los costes de personal, los costes de material y los costes indirectos obtenemos el presupuesto total del presente TFG y que observamos en la tabla A1.2.

Concepto	Presupuesto
Costes de material	4.335,98€
Costes de personal	20.800€
Costes indirectos (20%)	5.027,2€
Total	30.163,18€

Tabla A1.2: Costes totales

Apéndice 2: Cómo conectar la placa al ordenador y cómo cargar archivos en la placa

Lo primero doble clic al icono `Command Shell` (Ver figura A2.1).

Figura A2.1: Icono `Command Shell`

Sale la siguiente pantalla, que se puede ver en la figura A2.2. Se pone `smdetect` y se pulsa `enter`.

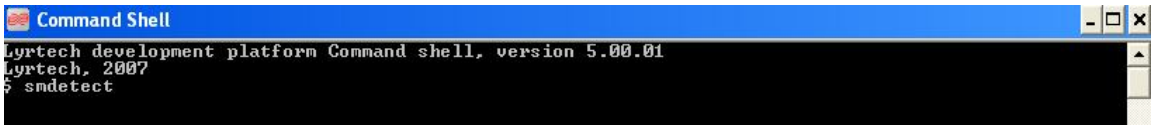

Figura A2.2: `Pantalla Command Shell`

La figura A2.3 muestra la pantalla para conectar la placa con el ordenador. En esta pantalla se selecciona `Enable WAN search` y se pone en `Start addres` 10.0.0.0 y en `Stop address` 10.0.0.10 para que encuentre a la placa. Se da al botón `Detect.` Se selecciona SDREVMD6446 (el del centro) y se pulsa el botón `Connect to Lyrtech development platform`.

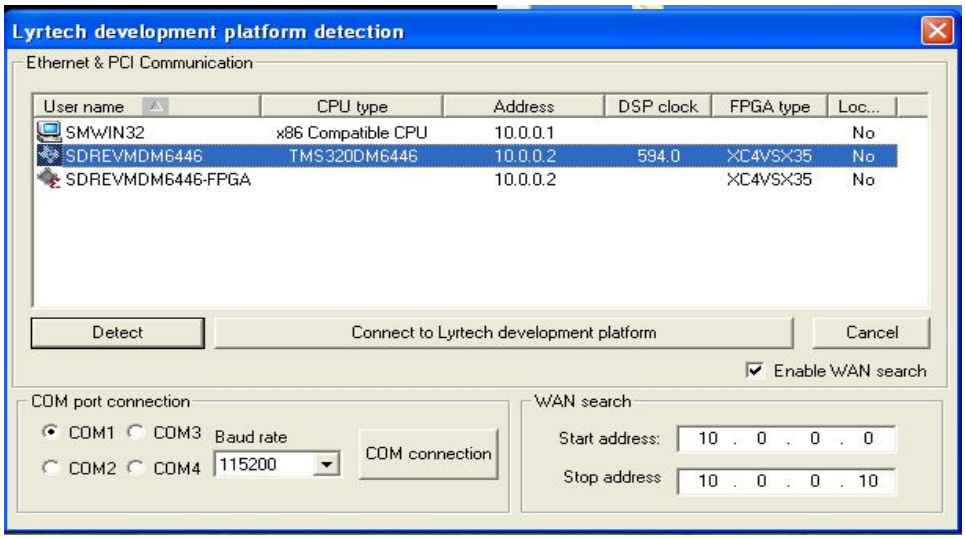

Figura A2.3: Pantalla de conexión placa-ordenador

Cuando ya está conectado. Sale el siguiente `Command Shell` en donde hay que cargar los archivos. (Ver figura A2.4)

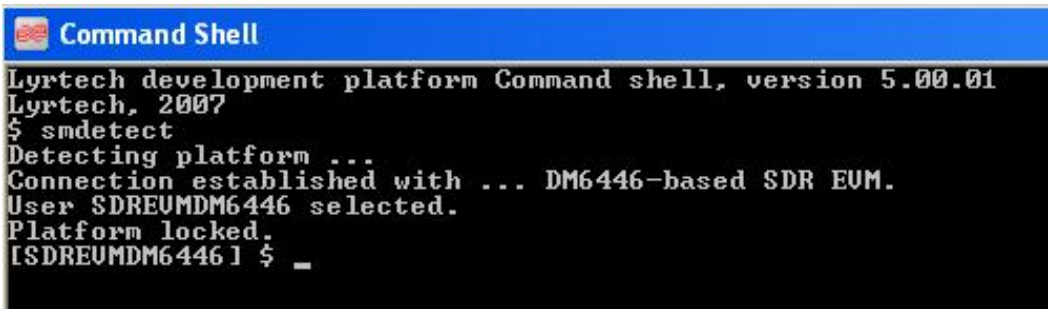

Figura A2.4: `Command Shell` para cargar los archivos

Para cargar archivos en la placa, primero se cargan los archivos del DSP y luego los de la FPGA. (Ver figura A2.5)

Hay que poner las siguientes líneas

`dspload` (y la ruta donde tengamos el archivo .out)

`fpgaload` (y la ruta donde tengamos el archivo .bit)

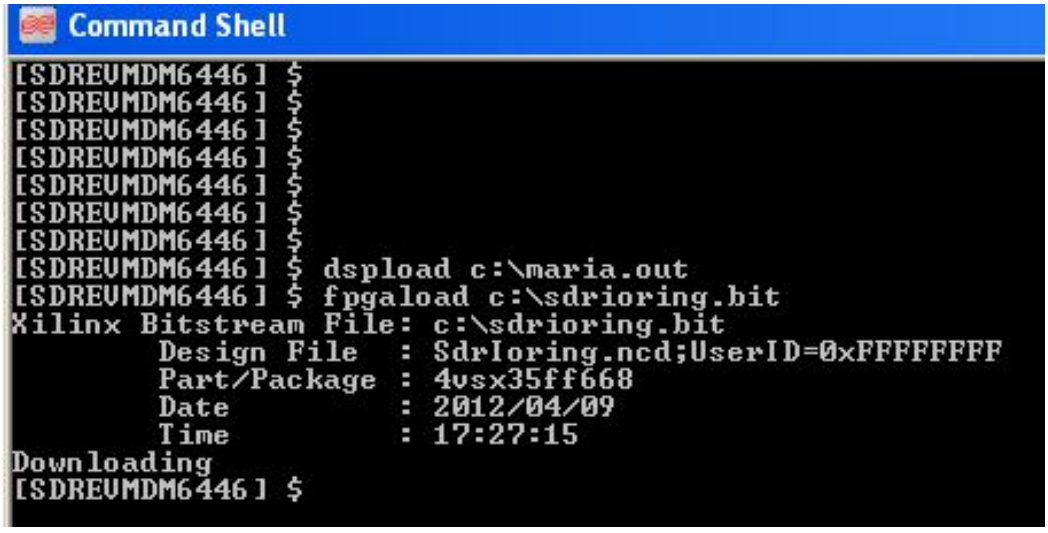

Figura A2.5: `Command Shell` archivos cargados

Apéndice 3: Instalación de Xilinx ISE

Cuando se comienza la instalación del programa Xilinx ISE, la primera pantalla que sale es la que aparece en la figura A3.1.

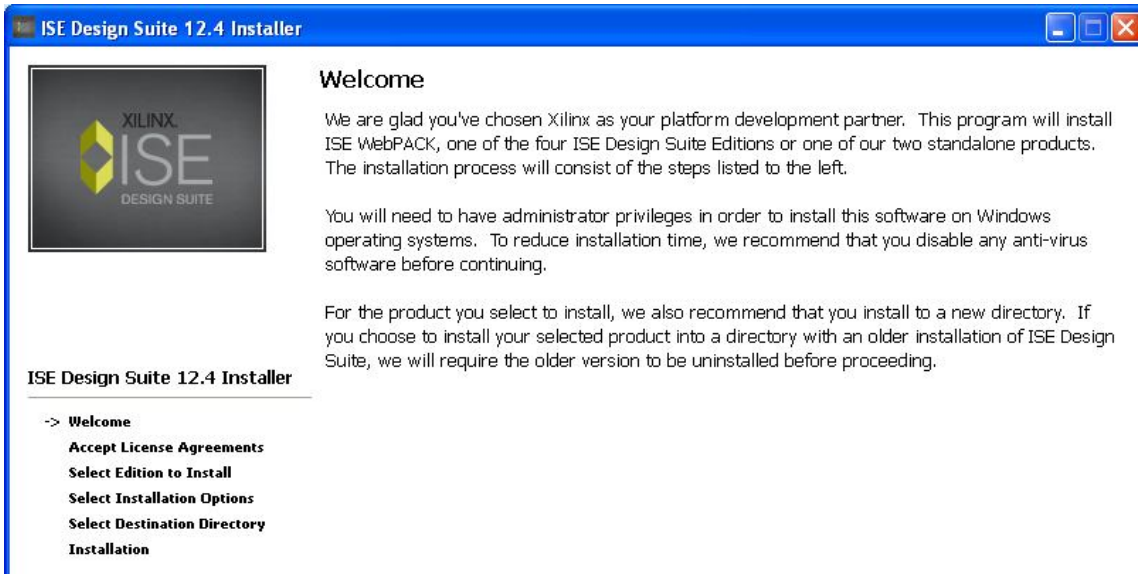

Figura A3.1: Primera pantalla de instalación de Xilinx ISE

Hay que aceptar la licencia. En la siguiente pantalla se selecciona `ISE DesignSuite: System Edition`. En las opciones, se seleccionan todas.

Hay que elegir el directorio donde se va a encontrar el programa. Se recomienda guardarlo en `C:\.` (Ver figura A3.2)

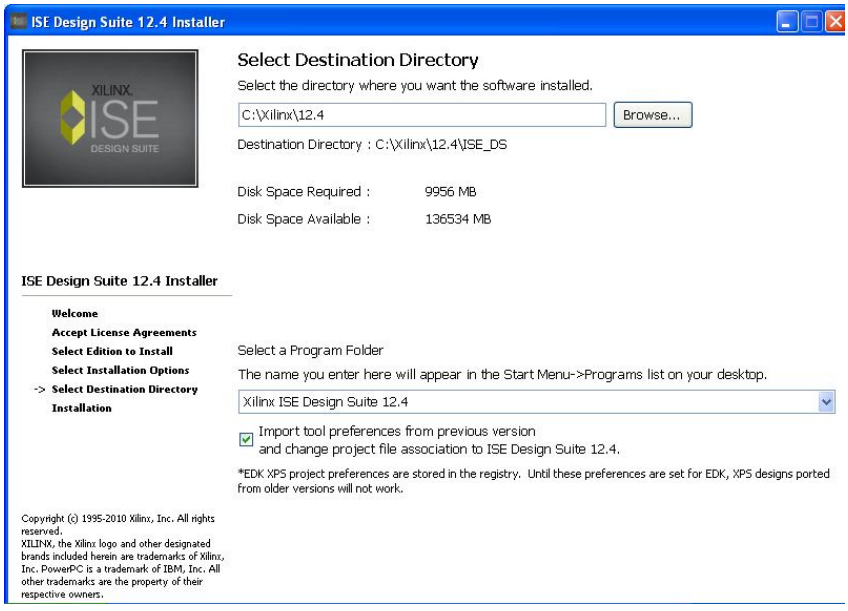

Figura A3.2: Directorio de Xilinx ISE

Y comienza la instalación.

Una vez hecho esto, aparece la siguente pantalla (ver figura A3.3), se selecciona, en la primera pestaña, `locate existing license(s)`.

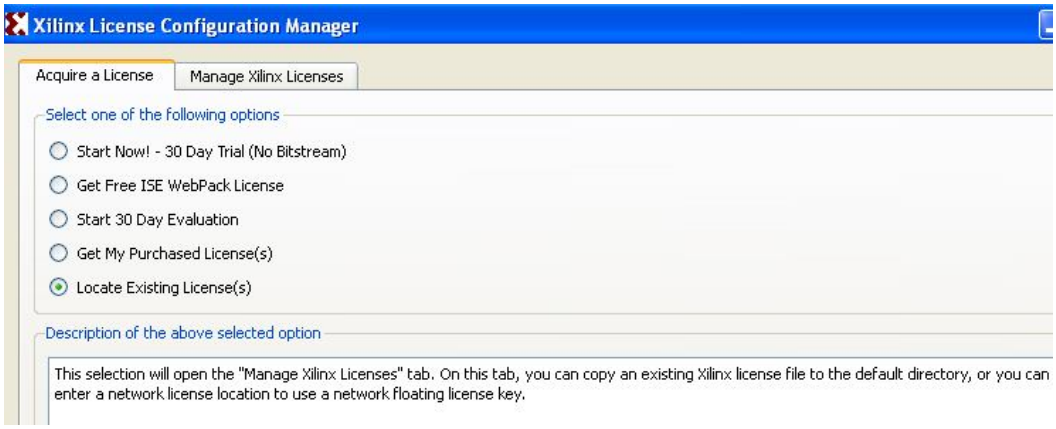

Figura A3.3: Pantalla de elección de la licencia

En la siguiente pestaña, en `XILINX_LICENSE_FILE` se pone la ruta en donde se ha guardado `license.lic`. Y en `LM_LICENSE_FILE`, se pone la ruta donde se ha puesto la licencia de Modelsim. (Ver figura A3.4)

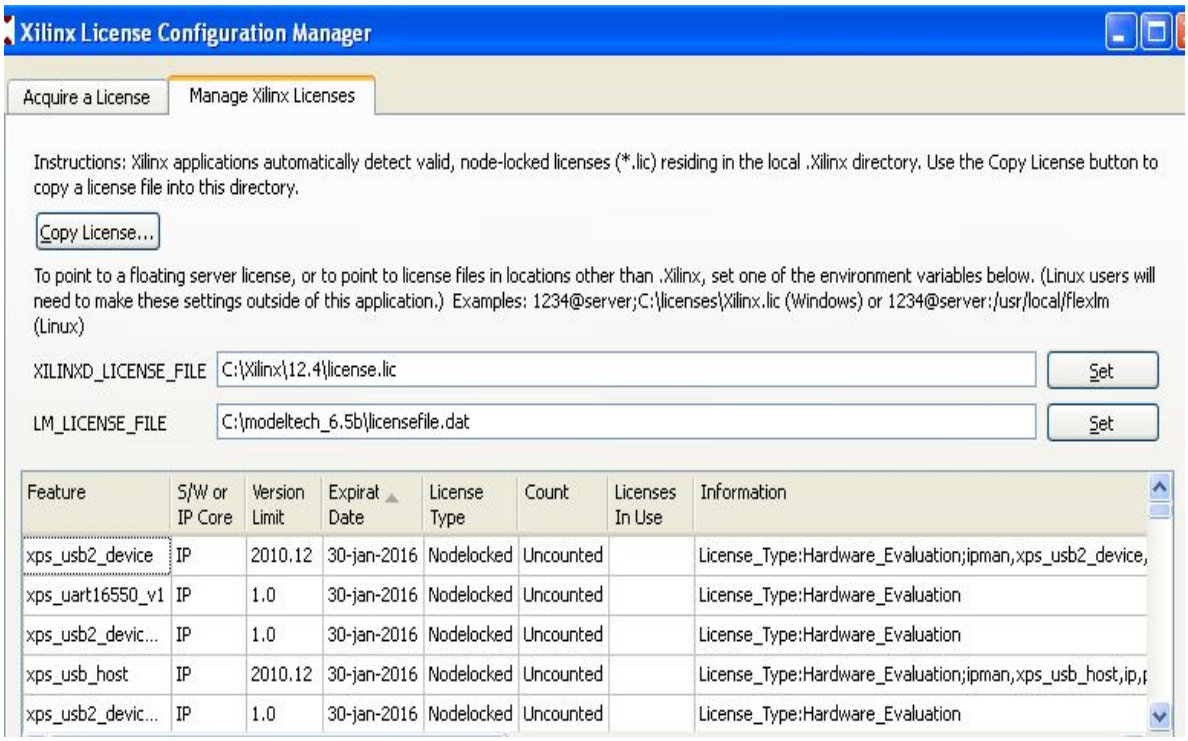

Figura A3.4: Pantalla de ruta de licencia

Se pulsa el botón `close` y ya esta.

Apéndice 4: Crear un nuevo proyecto con Xilinx ISE y compilar las librerías para poder simular

Primero hay que crearse un proyecto nuevo. Como se ve en la figura A4.1, se da al botón `New Project`.

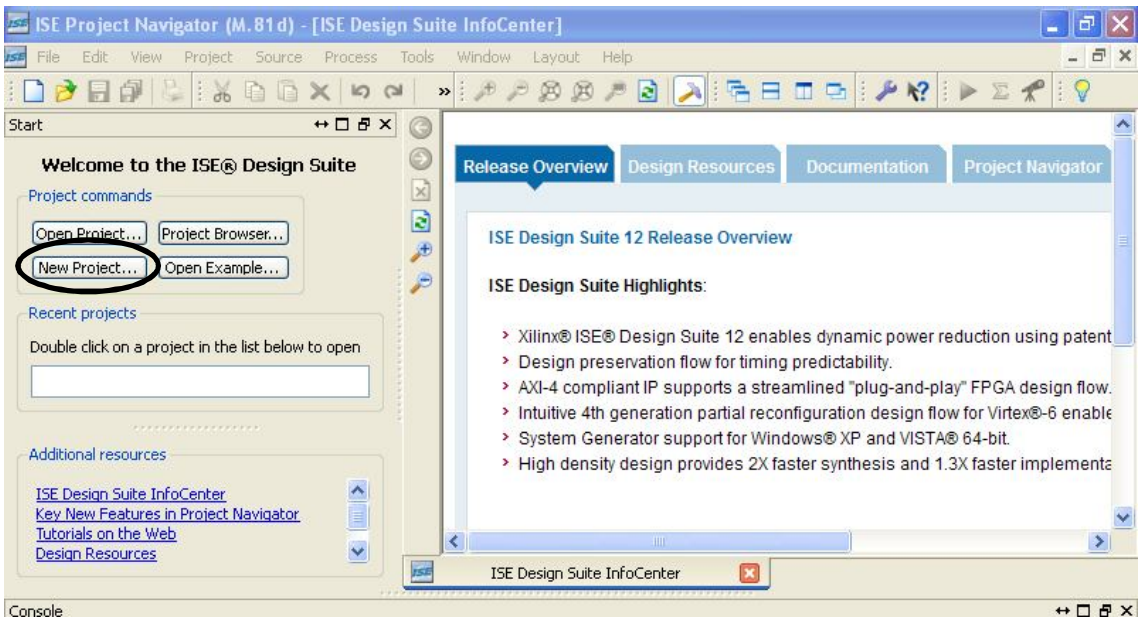

Figura A4.1: Pantalla de inicio de Xilinx ISE

Aparece la siguiente ventana, en donde se elige en nombre del proyecto con el que se va a trabajar y el directorio dónde se va a guardar. (Ver figura A1.2)

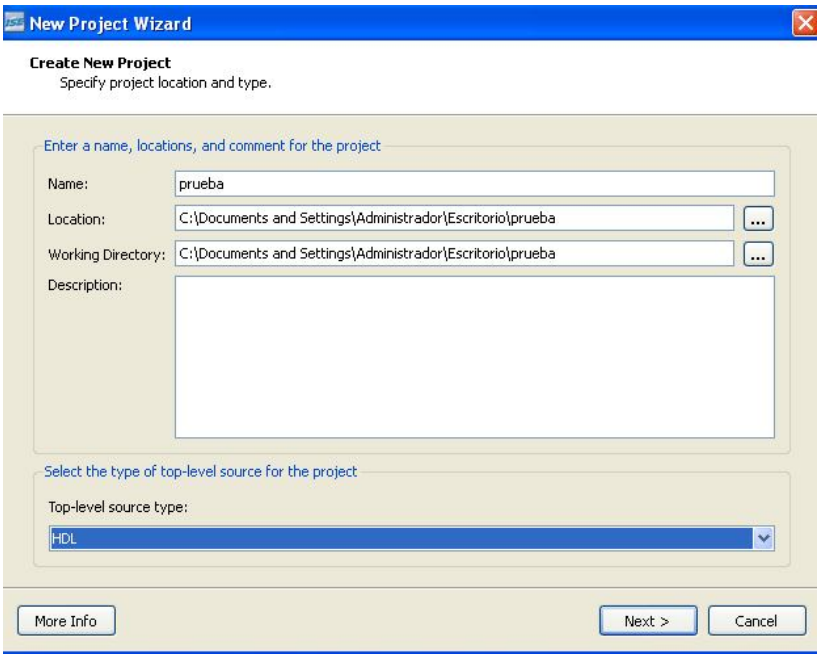

Figura A4.2: Pantalla de `New Project`

Se configura el proyecto con la FPGA con la que se va trabajar. Y se le indica donde se va a simular. Para utilizar esta placa, hay que poner lo que se ve en la figura A4.3.

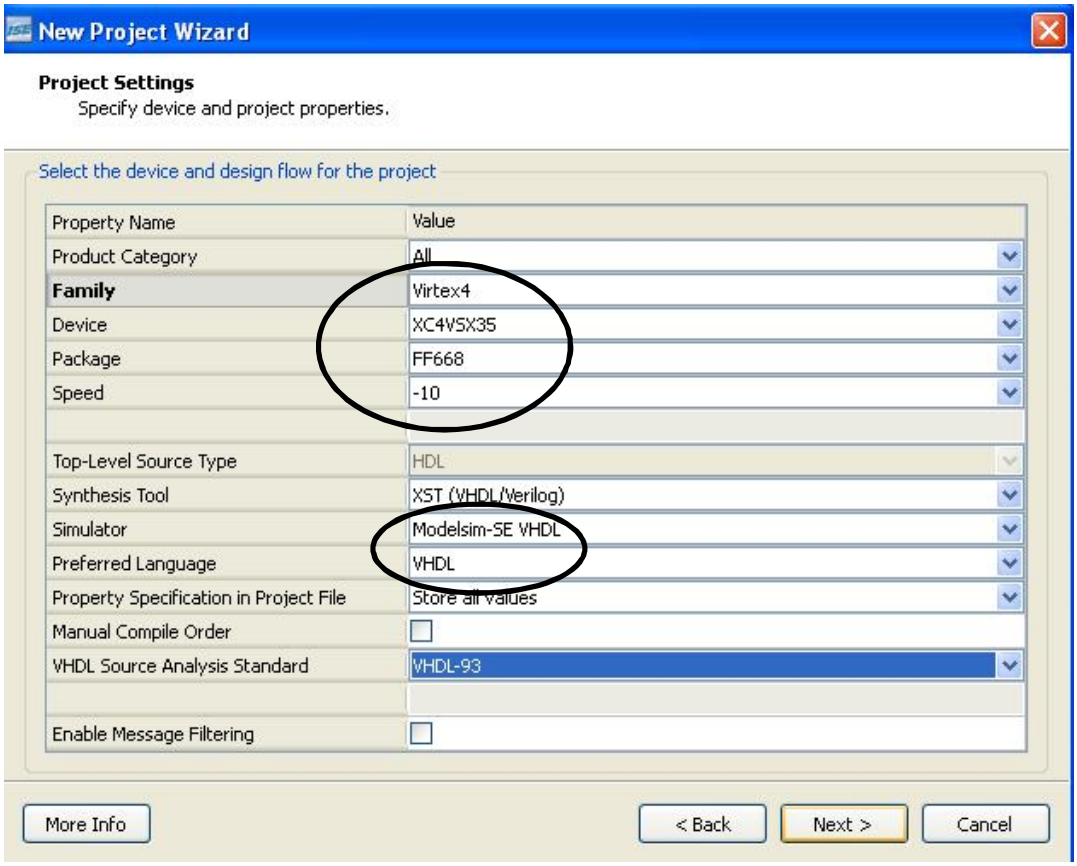

Figura A4.3: Configuración del proyecto en Xilinx ISE

Para añadir el código que se ha hecho en VHDL o los componentes de los IP Cores al proyecto, se da al menú `Project/Add Source…` (Ver figura A4.4)

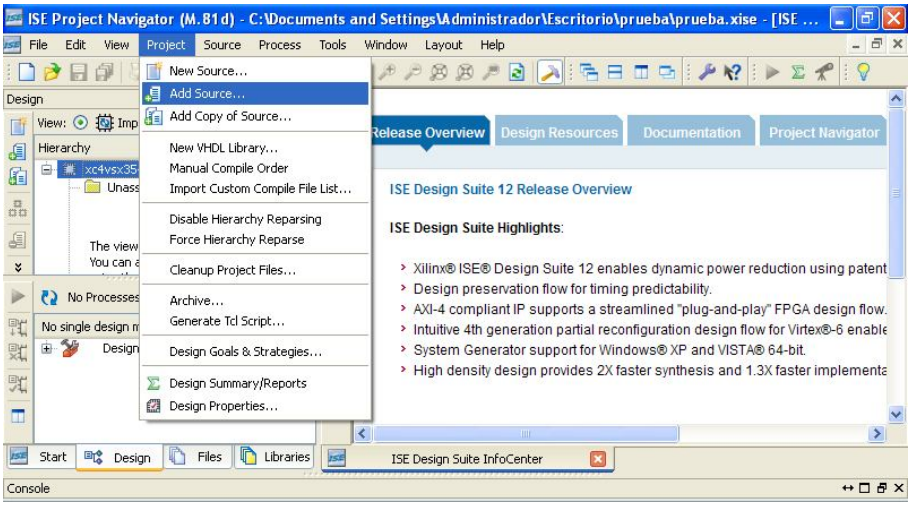

Figura A4.4: Añadir código al proyecto

Pero nada más terminar de instalar el Xilinx ISE, lo primero que hay que hacer es compilar las librerías.

Para compilar todas las librerías y poder simular en Modelsim, sin agregar nada al proyecto, se da a `Compile HDL Simulation Libraries/Run`(tarda bastante rato). (Ver figura A4.5)

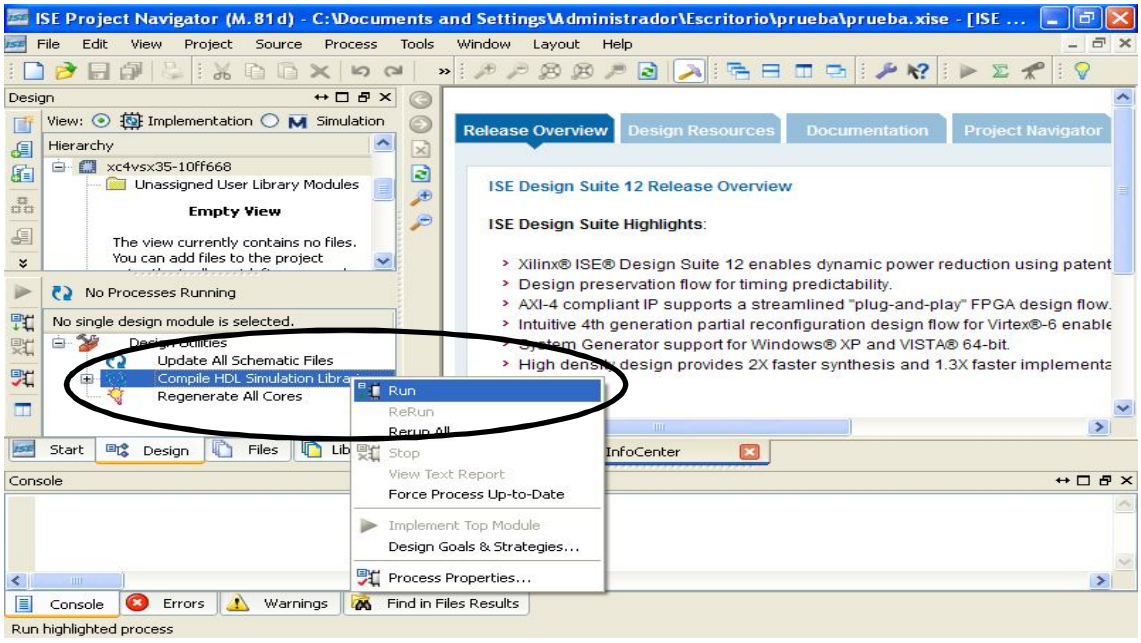

Figura A4.5: Compilar las librerías

El resultado de compilar las librerías es el que aparece en la figura A4.6.

```
*********************************************************************************************
*                                  COMPILATION SUMMARY                                      *
*                                                                                           *
*  Simulator used: mti_se                                                                   *
*  Compiled on: Sat Mar 10 13:25:38 2012                                                    *
*                                                                                           *
*********************************************************************************************
*              Library                |  Lang   |  Mapped Name(s)   | Err#(s)  |  Warn#(s)  *
*-------------------------------------------------------------------------------------------*
*  secureip                           | verilog | secureip          | 0        | 0          *
*-------------------------------------------------------------------------------------------*
*  unisim                             | vhdl    | unisim            | 0        | 0          *
*-------------------------------------------------------------------------------------------*
*  unisim                             | verilog | unisims_ver       | 0        | 0          *
*-------------------------------------------------------------------------------------------*
*  simprim                            | vhdl    | simprim           | 0        | 0          *
*-------------------------------------------------------------------------------------------*
*  simprim                            | verilog | simprims_ver      | 0        | 0          *
*-------------------------------------------------------------------------------------------*
*  xilinxcorelib                      | vhdl    | xilinxcorelib     | 0        | 238        *
*-------------------------------------------------------------------------------------------*
*  xilinxcorelib                      | verilog | xilinxcorelib_ver | 0        | 4          *
*-------------------------------------------------------------------------------------------*
*  edk                                |         | edk               | 0        | 156        *
*-------------------------------------------------------------------------------------------*
```

Figura A4.6: Resultado de compilar las librerías.

Apéndice 5: Instalación de Modelsim

Lo primero se hace doble clic al icono `modelsim-win32-6.5b-se Mentor Install Bundle`. (Ver figura A5.1)

Figura A5.1: Icono `Modelsim Install`

Aparecerá la siguiente ventana que se muestra a continuación en la figura A5.2. Se pulsa `Next`.

Figura A5.2: Inicio de la instalación de Modelsim

A continuación aparece una ventana en donde se acepta la licencia. Se pulsa el botón `Agree.`

Se indica donde se va a guardar la carpeta de Modelsim. Se recomienda ponerlo directamente en `C:\.` Le damos a `Next.` (Ver figura A5.3)

Figura A5.3: Directorio de Modelsim

Dirá que no existe el directorio. Se pulsa a `Yes` para que cree el directorio de modeltech_6.5b. (Ver figura A5.4)

Empieza la instalación.

Se pulsa a `Yes` para que cree un acceso directo en el escritorio.(Ver figura A5.5)

Figura A5.4: Crear directorio de modeltech_6.5

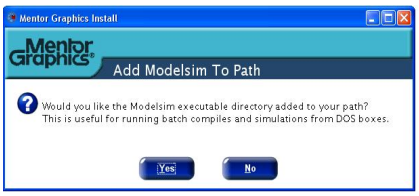

Figura A5.5: Crear acceso directo de Modelsim en el escritorio

Se pulsa `Yes` para que nos instale unos `drivers` (lo recomienda).

Para que termine de instalar el programa, hay que reiniciar el ordenador.

Una vez el ordenador se haya reiniciado, hay que ir a Mi PC. Con el ratón se pincha, en cualquier lugar, con el botón derecho/propiedades.

En propiedades del sistema/opciones avanzadas/variables de entorno.(Ver figura A5.6)

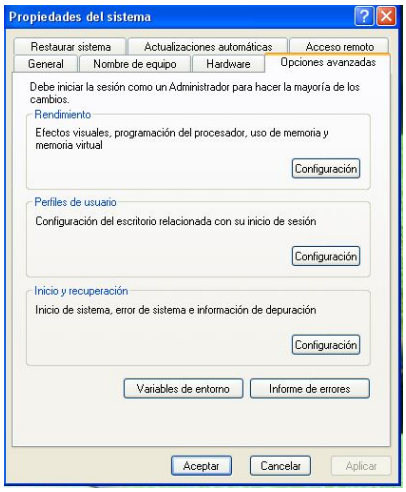

Figura A5.6: Ventana de propiedades del Sistema

En variables del sistema, se crea una nueva. Como se aprecia en la figura A5.7, hay que escribir `LM_LICENSE_FILE` en el Nombre de la variable y `C:\modeltech_6.5b/licensefile.dat` en el Valor de la variable.

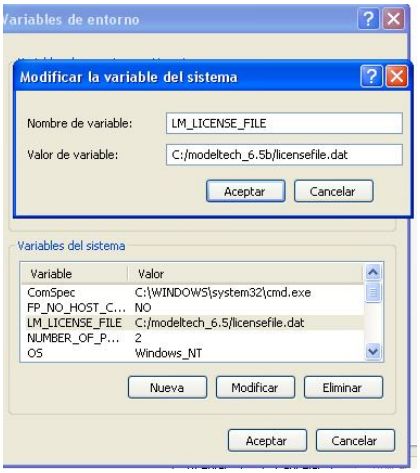

Figura A5.7: Variables del sistema

Se copia la carpeta Xilinx en `C:\`, para poder compilar y simular los programas que vaya ha utilizar para la placa. (Ver figura A5.8)

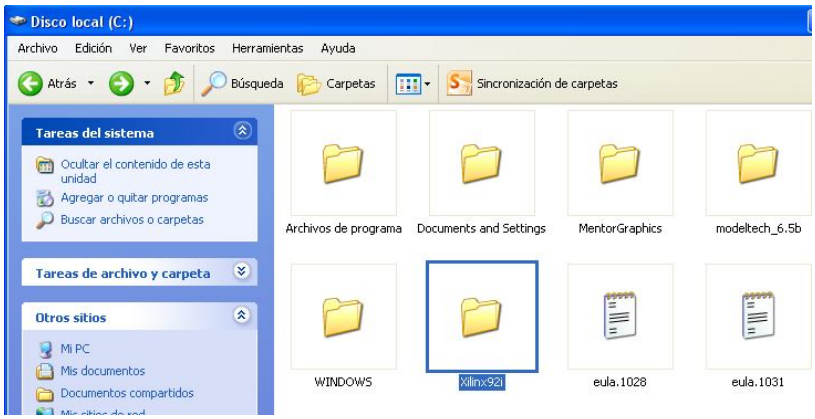

Figura A5.8: Carpeta Xilinx

Una vez hecho esto, en c:\modeltech_6.5b, se da doble clic en modelsim (opciones de configuración). (Ver figura A5.9)

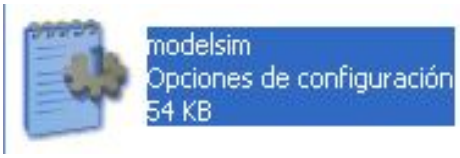

Figura A5.9: Opciones de configuración de Modelsim

Si está protegido de sólo lectura, se desactiva y aceptar.

Se abre un blog de notas, y se añaden las librerías de Xilinx. (Ver figura A5.9)

```
SIMPRIM =
C:\Xilinx\12.4\ISE_DS\ISE\vhdl\mti_se\6.5b\nt\simprim

UNISIM =
C:\Xilinx\12.4\ISE_DS\ISE\vhdl\mti_se\6.5b\nt\unisim

XILINXCORELIB =
C:\Xilinx\12.4\ISE_DS\ISE\vhdl\mti_se\6.5b\nt\xilinxcorelib
```

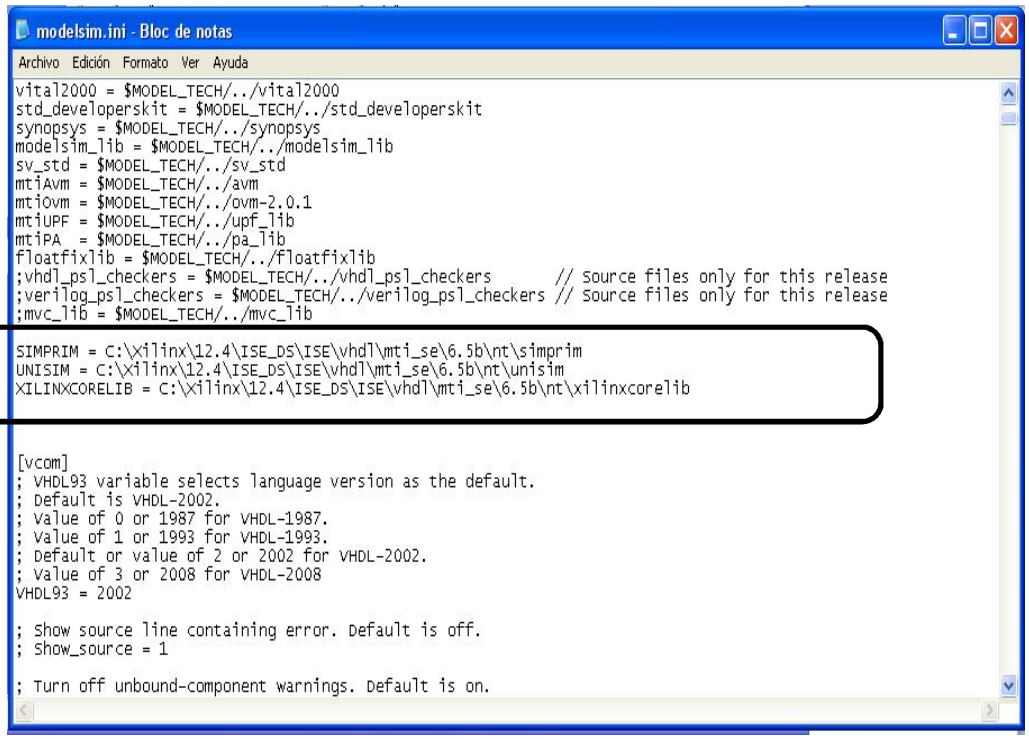

Figura A5.9: Modelsim.ini

Está listo para empezar a trabajar en Modelsim.

Apéndice 6: Simular en Modelsim

Se va a simular el seno que se ha generado con los `IP Core`, en Modelsim. Con esto se comprueba si funcionaria o no en la placa.

En la figura A6.1, se muestra la pantalla principal de Modelsim. Lo primero de todo se crea un nuevo proyecto, `File/New/Project.`

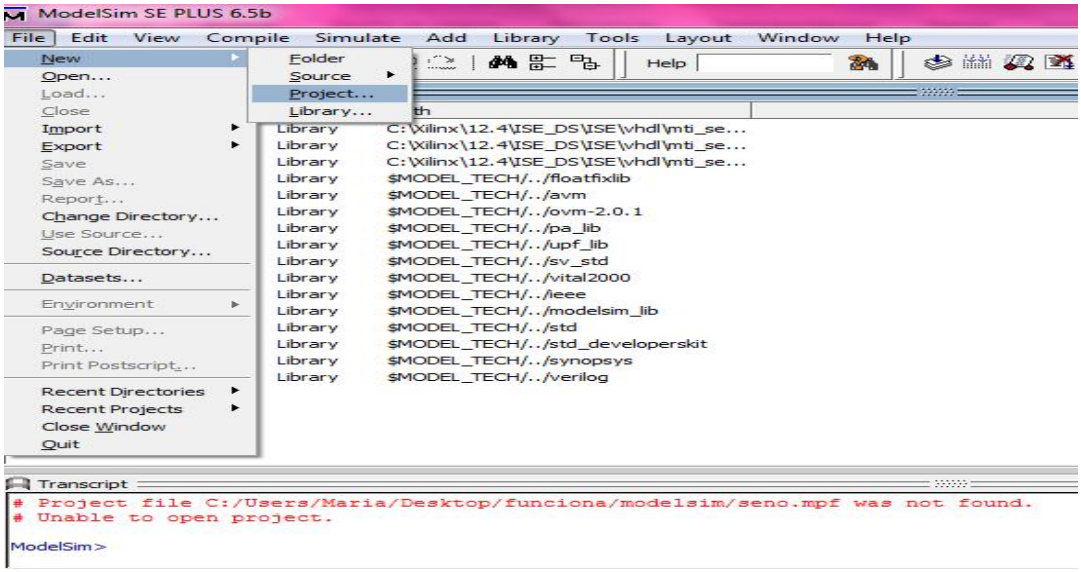

Figura A6.1: Pantalla principal de Modelsim.

Se le da un nombre al proyecto y se guarda en una carpeta.

Para añadir archivos al proyecto se pulsa a `Add Existing file` y se seleccionan todos los archivos que vayan en el proyecto. (Ver figura A6.2)

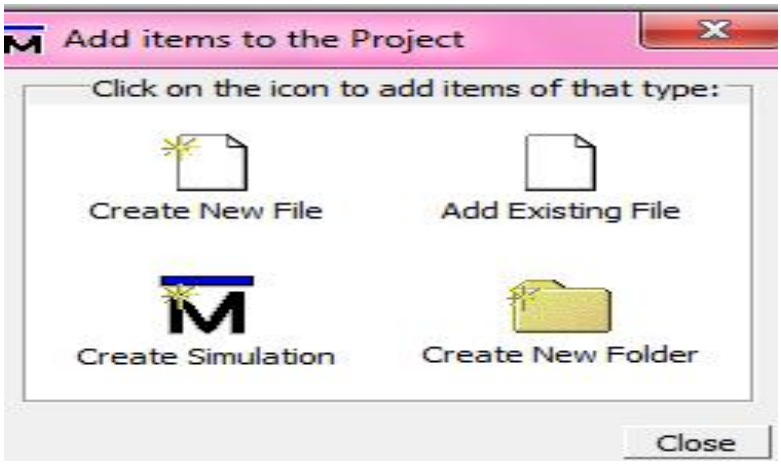

Figura A6.2: Añadir archivos al proyecto.

Una vez se tengan todos los archivos agregados a al proyecto hay que compilar. Con el ratón se da al botón derecho sobre un archivo, `Compile/Compile All`. Se puede ver en la figura A6.3.

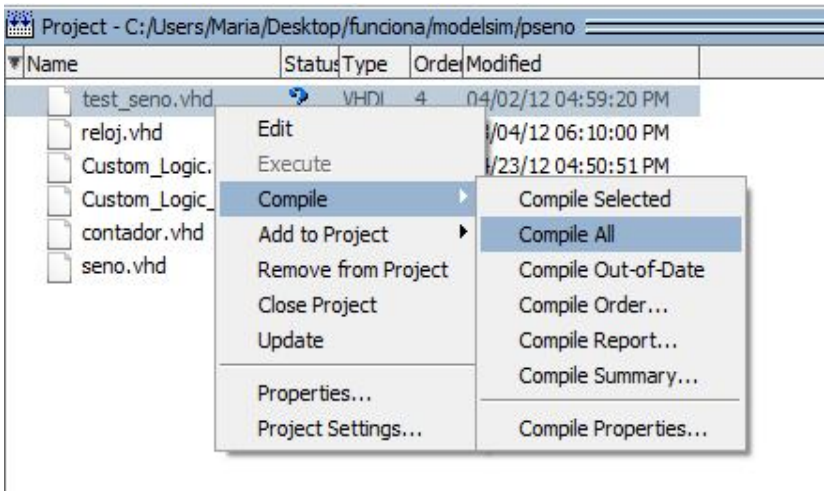

Figura A6.3: Compilar los archivos del proyecto.

Una vez que todo compile bien, en la pestaña de `Library,` se despliega la carpeta `work`. En `test_seno`, que es el programa para probarlo, con el botón derecho del ratón/`Simulate without Optimization`.(Ver figura A6.4)

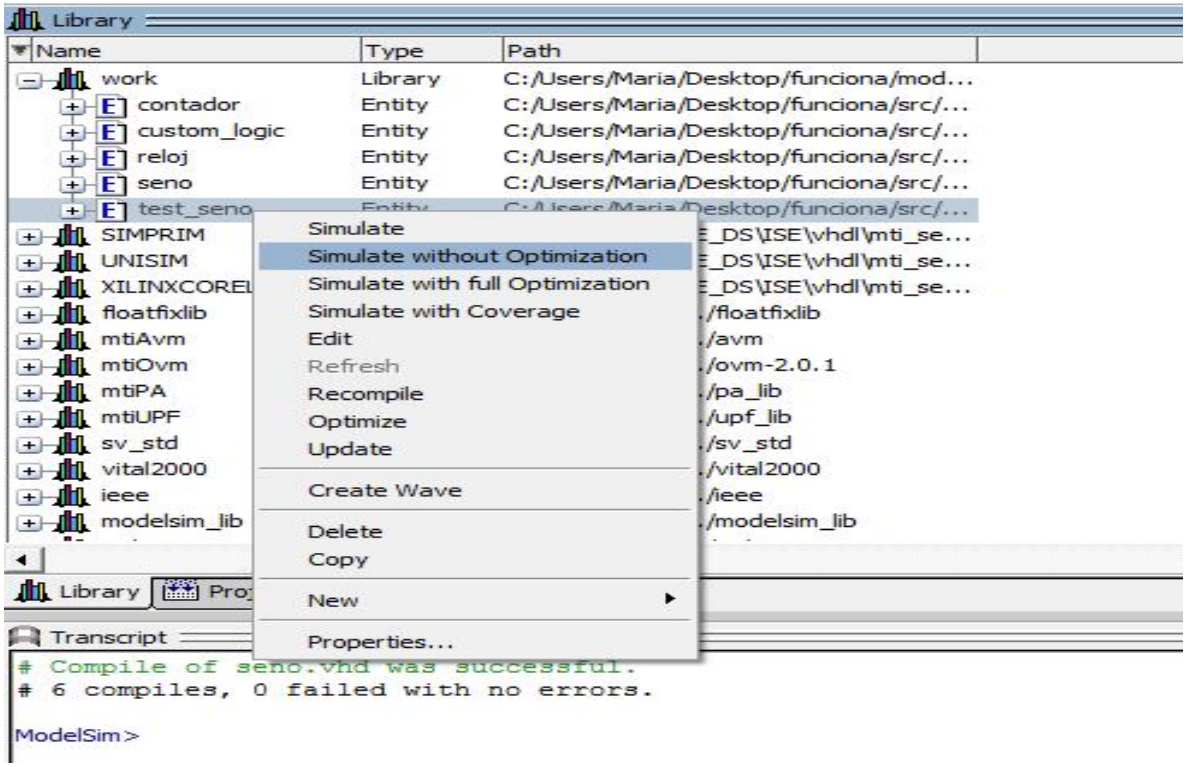

Figura A6.4: Pestaña `Library` para simular en Modelsim

Cuando acabe de simular, saldrá la siguiente pantalla que se muestra en la figura A6.5.

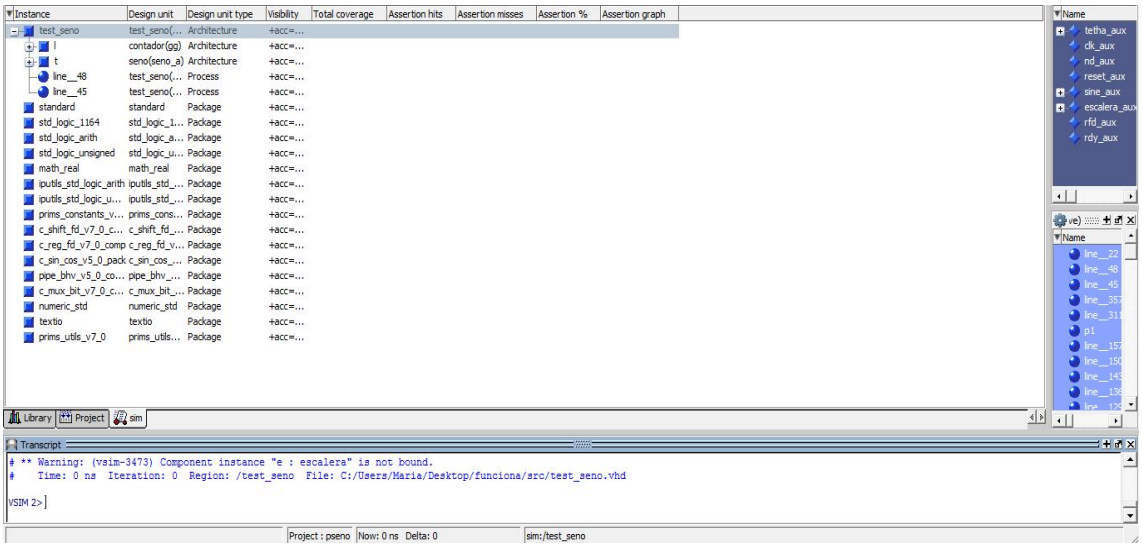

Figura A6.5: Pantalla pre-simulación de Modelsim

Para que salga la página de simulación llamada Wave, con el botón derecho del ratón sobre `test_seno/Add/To Wave/All ítems in región and below.`

Cuando salga la página Wave, le doy a Run.(Ver figura A6.6)

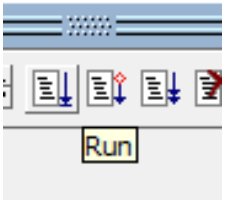

Figura A6.6: Botón de Run

Aparecerán las señales de todas las variables.

Para ver el seno en forma de seno y no en bits le damos al botón derecho del ratón en la variable seno/`format/analog.` (Ver figura A6.7)

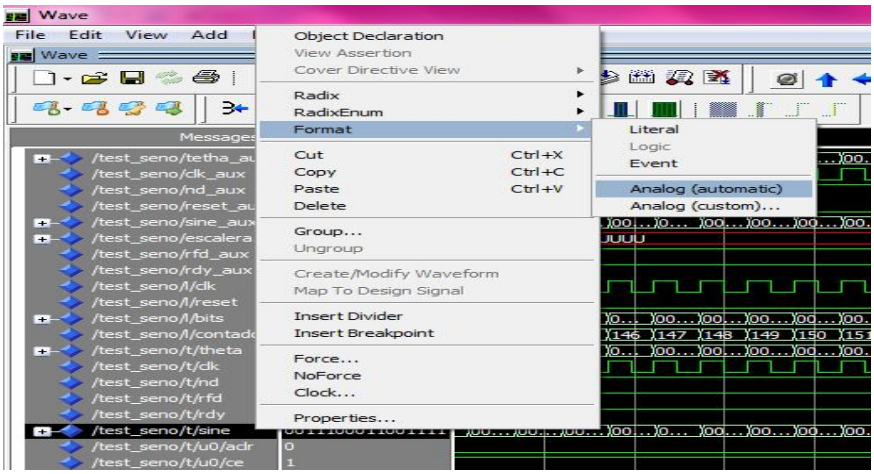

Figura A6.7: Pasar el seno de bits a forma analógica.

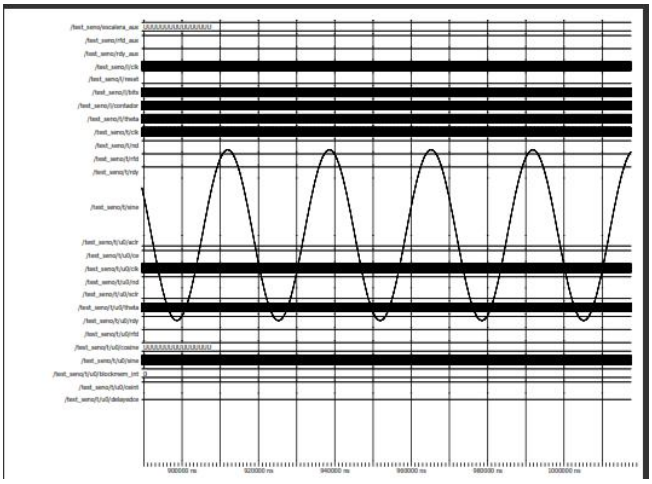

Figura A6.8: Simulación en forma analógica.

Como se puede ver en la figura A6.8 ya lo está simulado. Se puede saber la frecuencia del seno, amplitud,…

Apéndice 7: Configuración del Code Composer Studio para trabajar con el DSP de la placa.

Una vez se haya instalado el CCS, se da doble clic a `Setup CCStudio v3.3`. (Ver figura A7.1)

Figura A7.1: `Setup CCStudio v3.3`

Aparece la siguiente ventana mostrada en la figura A7.2.

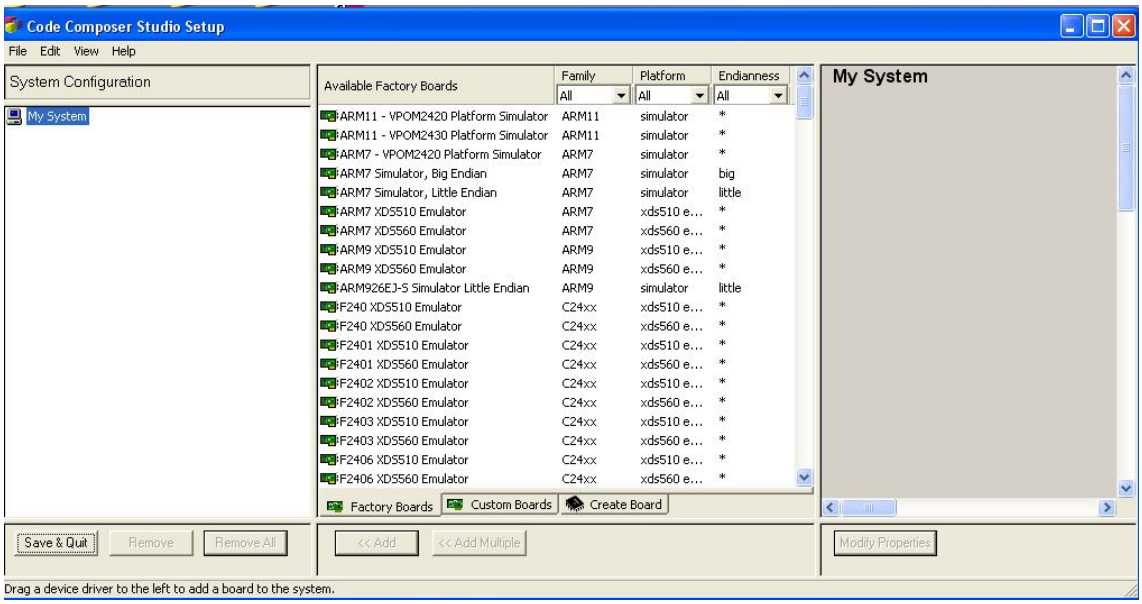

Figura A7.2: Pantalla de configuración del CCS.

En `Family` se selecciona `C64x+`, en `Platform` se selecciona `Simulator`, y una vez elegido esto aparece la siguiente ventana mostrada en la figura A7.3. Se selecciona `DM6446 Cycle Accurate Simulator`

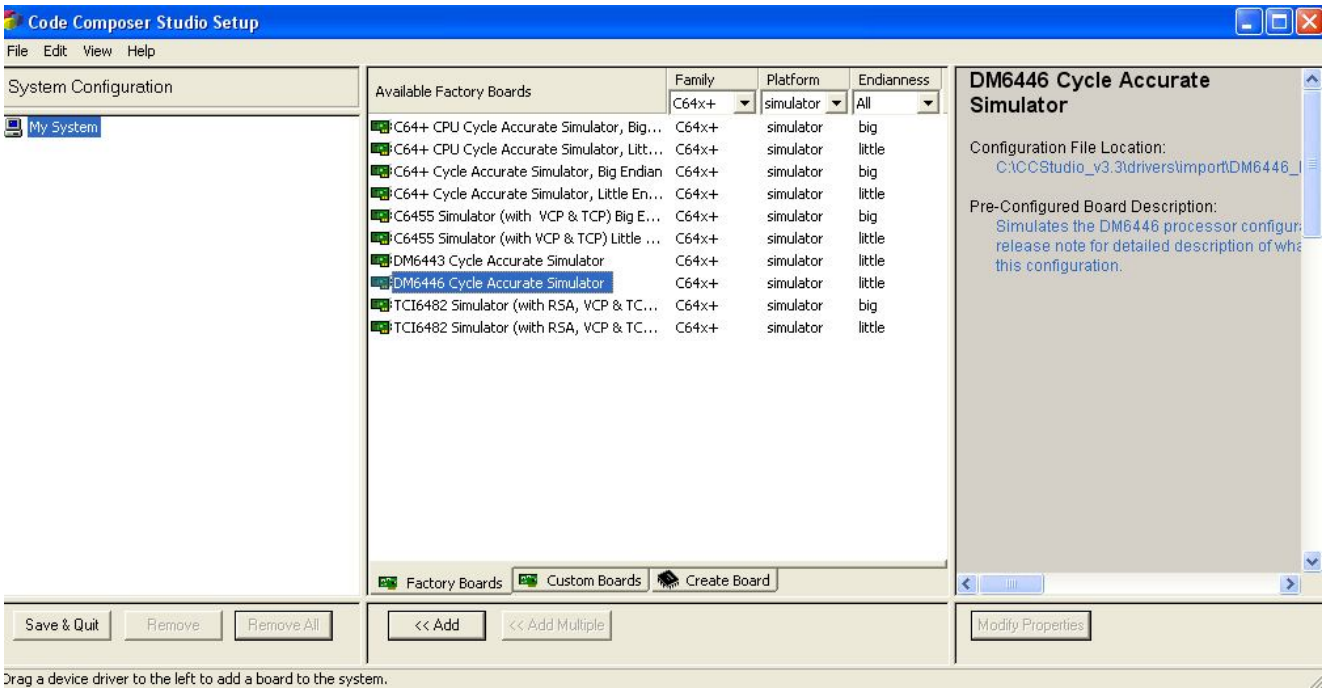

Figura A7.3: Selección del DSP

Se da doble clic y ya está configurado el CCS para trabajar con el DSP del dispositivo.

Apéndice 8: Cómo se crea un proyecto en el Code Composer v3.3

Para crear un proyecto en el CCS, lo primero se recomienda crear una carpeta donde se vaya a guardar el proyecto. Dentro de esa carpeta hacer otras tres carpetas para guardar el código que se ha programado (src), otra para guardar el proyecto (pjt) y otra para guardar la configuración (conf). (Ver figura A8.1)

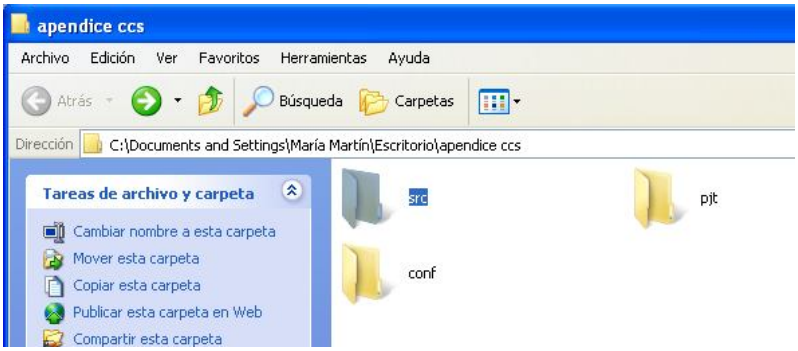

Figura A8.1: Carpeta del proyecto del CCS

Se abre el programa. Se crea un nuevo proyecto, para ello hay que ir al menú `Project/New Project`. En esta ventana, que se muestra en la figura A8.2, se pone el nombre del proyecto, la carpeta donde se crea el proyecto (pjt), el tipo de proyecto (*.out) y el modelo de la tarjeta DSP.

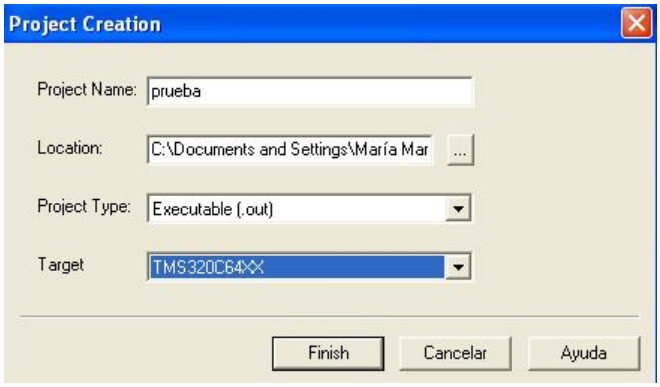

Figura A8.2: Ventana de creación del proyecto en CCS

Para añadir el código al proyecto `Project/Add files to Project`. Y se eligen los archivos que se necesiten. (Ver figura A8.3)

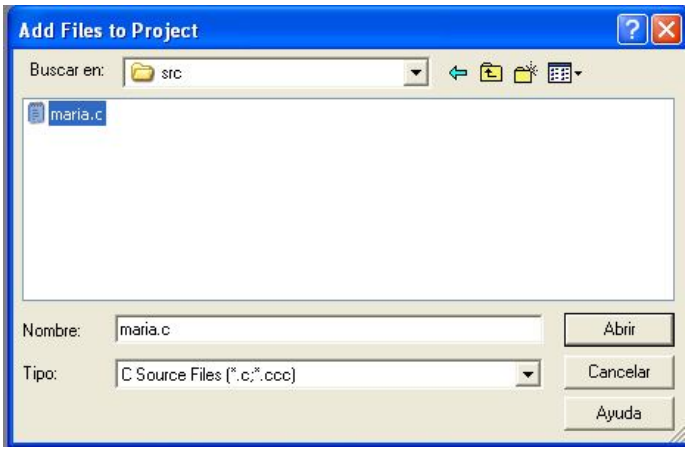

Figura A8.3: Añadir archivos al proyecto de CCS

Ahora se necesita agregar las librerías y archivos de cabecera. Se deben agregar los directorios al proyecto para que incluya las cabeceras, ya que sino se hace, el compilador no es capaz de encontrar los archivos mencionados en `#include`. Esto se hace en `Project/Build Options`.

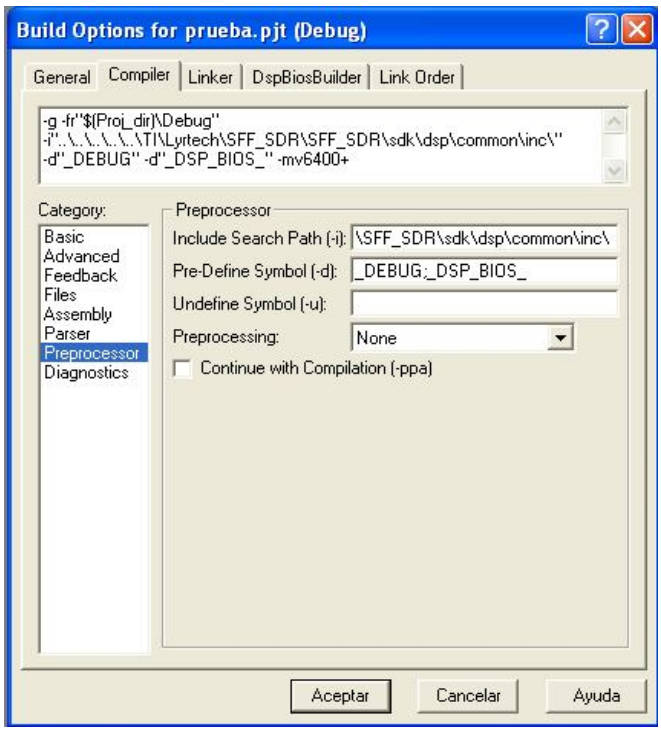

Figura A8.4: `Build Options` pestaña `Compiler`

En la figura A8.4 se muestra la pestaña `Compiler`, se selecciona la categoría `Preprocesor` y se ingresa en `Include Search Path (i)` el `path` donde se encuentren los `include`.

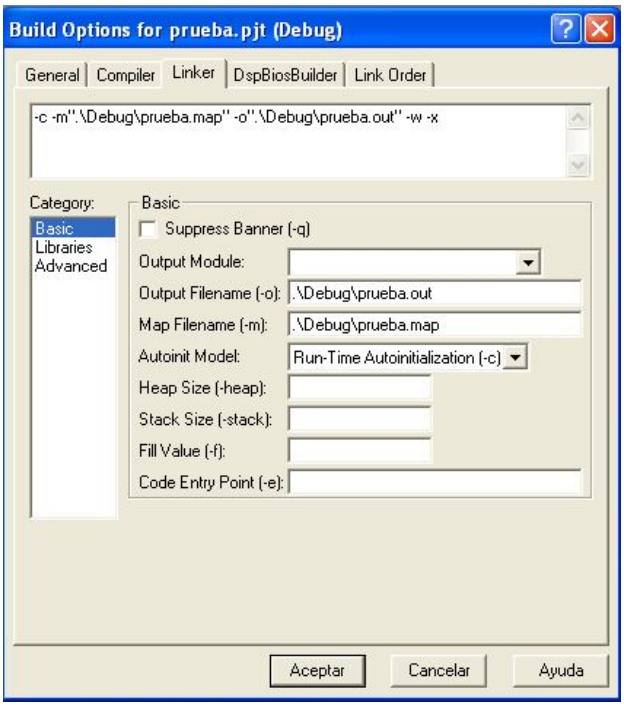

Figura A8.5: `Build Options` pestaña `Linker`

En la pestaña `Linker`, ingresar en la categoría `Basic` el nombre del archivo *.out. (Ver figura A8.5)

Una vez se tengan las librerías y los includes, hay que configurar la memoria con los archivos .cmd. En el menú `Project/add files to Project` se selecciona el archivo .cmd

Una vez hecho esto, se da a `Project/Rebuilt all`, y cuando compile todo bien, se habrá generado el archivo .out para cargarlo en la placa.

Apéndice 9: Generar un seno con IP Core.

Los IP Core son bloques de código HDL que otros ingenieros ya han escrito anteriormente para llevar a cabo una función específica. A pesar de que puede simplificar un determinado diseño, hay que diseñar las interfaces para enviar y recibir datos de esta “caja negra”. Ver en bibliografía [19].

Se abre el `CORE Generador`, `Inicio/Programas/Xilinx ISE/ISE Design Tools/Tools/CORE Generator.`

Una vez abierto, se crea un nuevo proyecto, `File/New Proyect.` (Ver Figura A9.1)

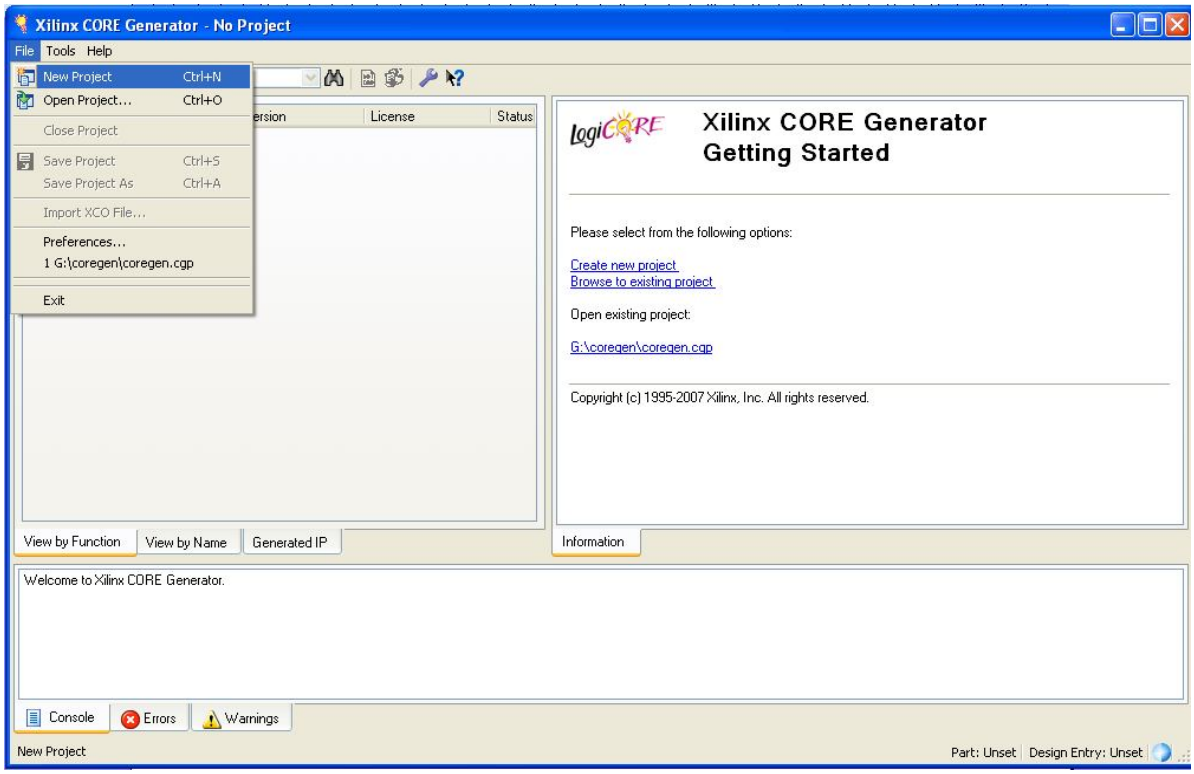

Figura A9.1: Ventana principal `CORE Generator`

Se da nombre al proyecto, y se indica donde se tiene que guardar.

Se configura el IP Core que se va a utilizar, que en este caso es un seno.

En la figura A9.2, se muestra la primera pestaña en donde se configura la FPGA. En `Family` se pone `Virtex4` ya que la FPGA de la placa es una `Virtex4`, en `Device` se selecciona `xc4vsx35`. En el `Package` se elige `ff668` y en el `Speed Grade` -10.

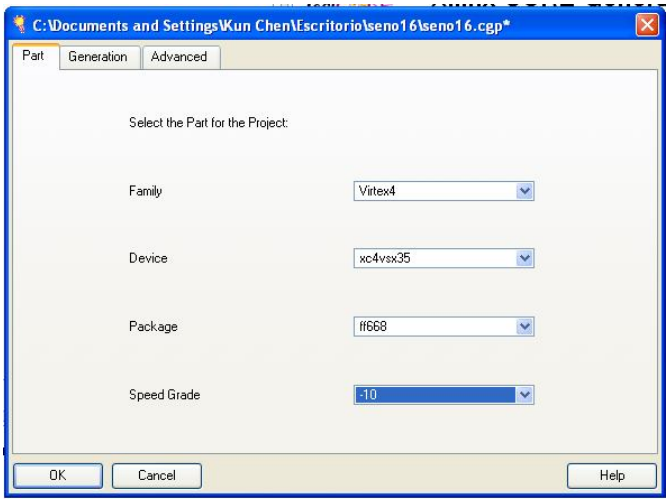

Figura A9.2: Pestaña `Part` de la ventana de configuración del dispositivo

La segunda pestaña, `Generation` se deja como viene por defecto. (Ver figura A9.3)

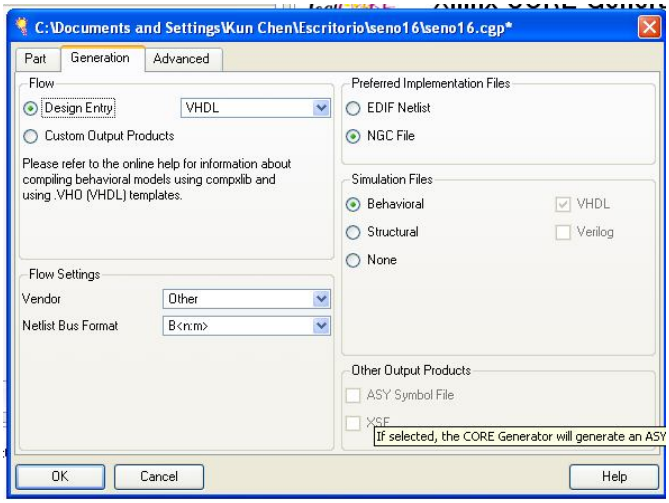

Figura A9.3: Pestaña `Generation` de la ventana de configuración del dispositivo

En la tercera pestaña, Advanced, se selecciona la primera opción (`create netlist Wrapper with IO paths`). (Ver figura A9.4). Se pulsa `OK`.

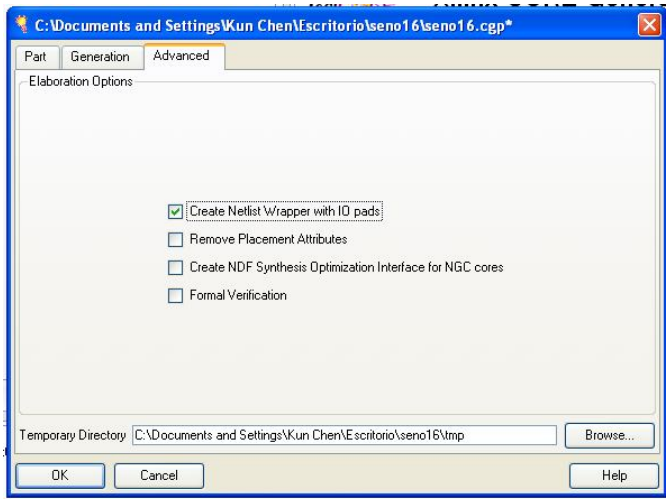

Figura A9.4: Pestaña Advanced de la ventana de configuración del dispositivo

En la figura A9.5 se muestra la pantalla principal de CORE Generator. En esta pantalla se puede elegir la función del IP Core que se necesite.

Se elige en las jerarquías, en este caso, Digital Signal Processing/Trig Functions/Sine-Cosine y se da a Customize. Para configurar esta parte es recomendable mirar en las hojas de características para que cada uno configure lo que necesita.

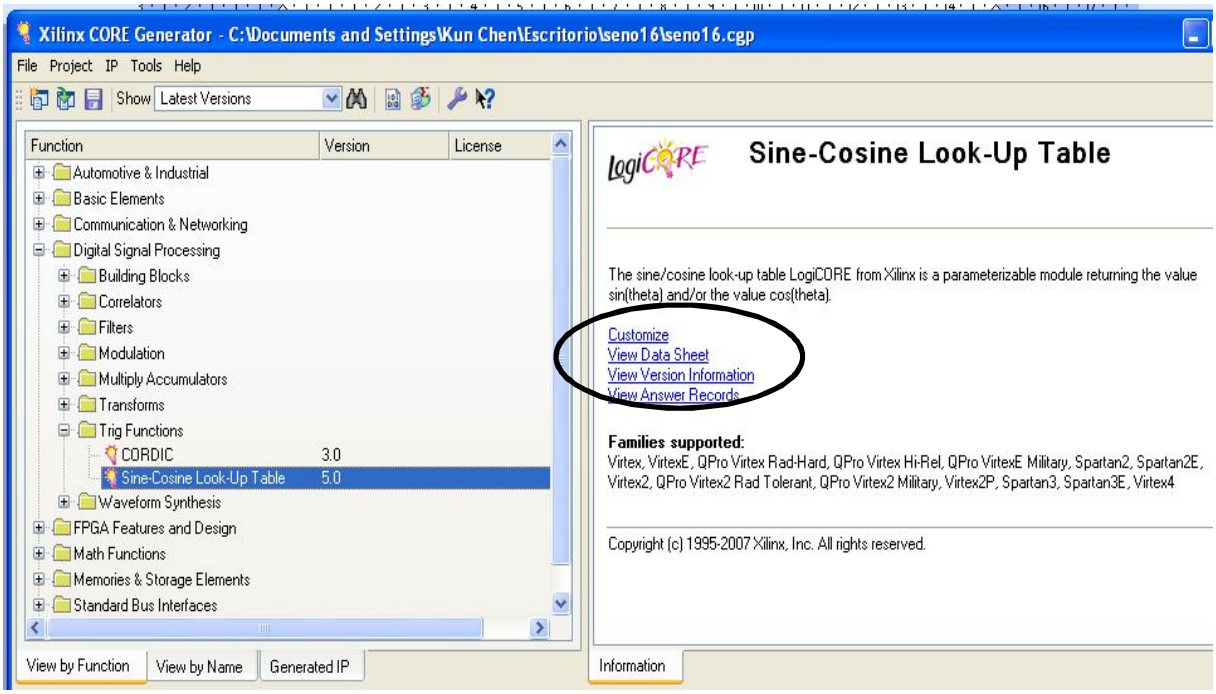

Figura A9.5: Ventana principal CORE Generator para elegir IP Core

En la pestaña `Parameters` (Ver figura A9.6), se pone el nombre del componente, el ancho del seno y el valor de `Theta` dependiendo de cuanto de preciso se quiere el seno (en este caso 16bits), se elige la función (seno) y el tipo de memoria. Se pulsa `Next` hasta llegar a la última página, dejando los valores que vienen por defecto. (Ver figura A9.7, A9.8)

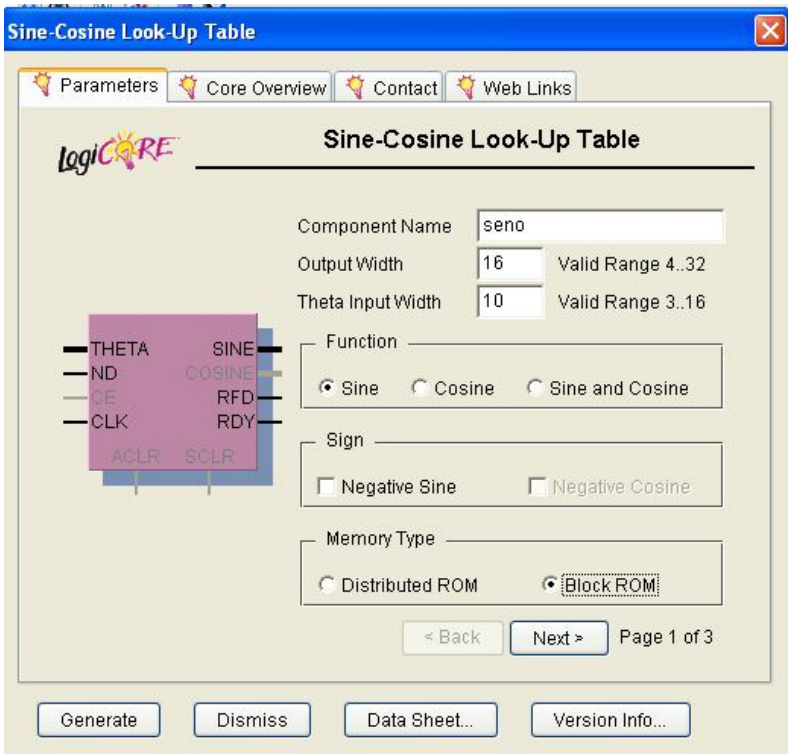

Figura A9.6: Pestaña `Parameters` de configuración del seno

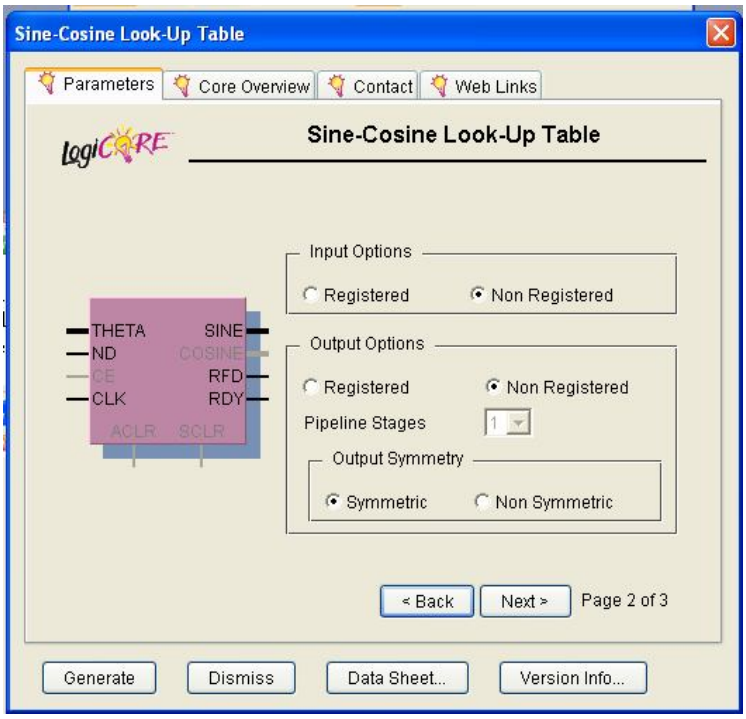

Figura A9.7: Pestaña `Parameters` de configuración del seno

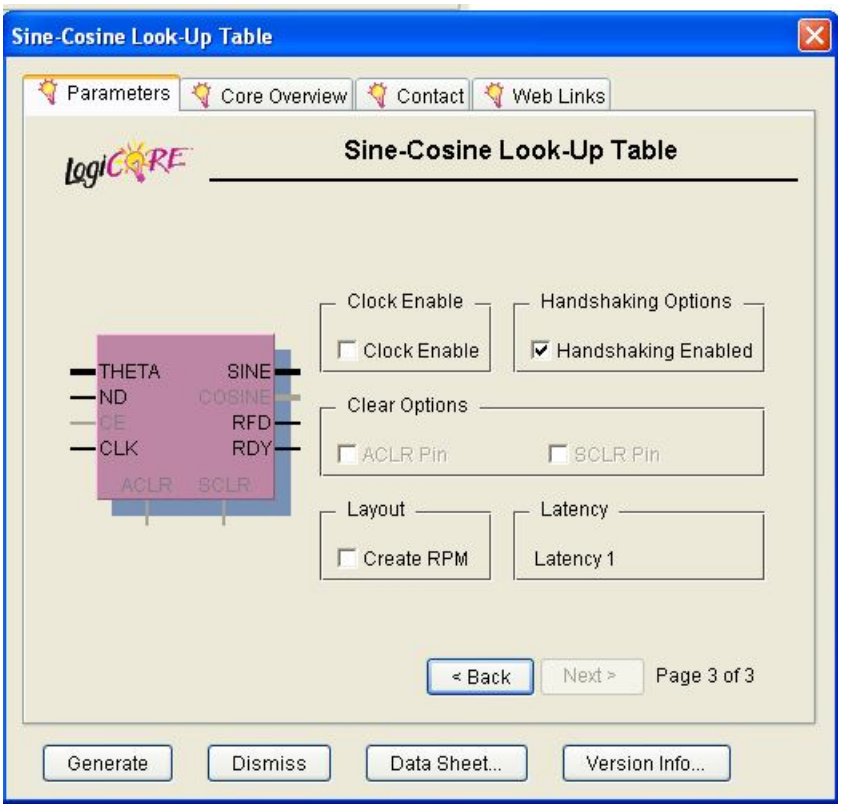

Figura A9.8: Pestaña `Parameters` de configuración del seno

Cuando ya esta todo configurado se pulsa a generar.

Ya está el seno generado con los IP Core. (Ver figura A9.9)

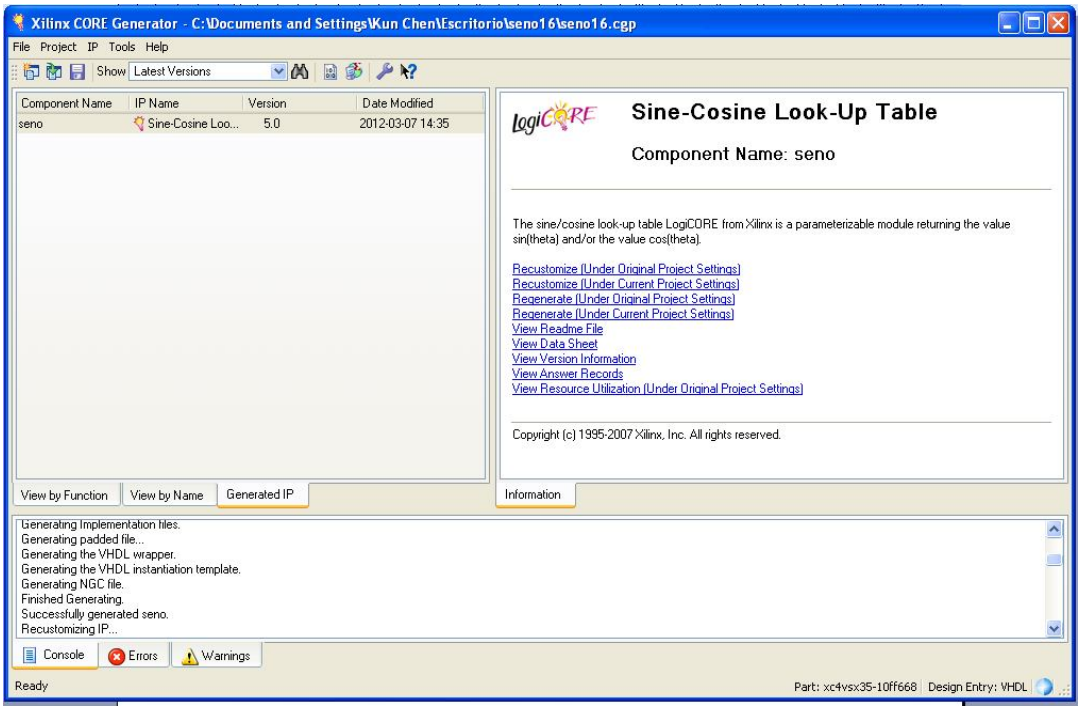

Figura A9.9: Seno generado con IP Core

8 BIBLIOGRAFÍA

[1] Lyrtech Inc. “SFF SDR Development Platform User’s guide” version 1.1, Octubre de 2008.

[2] Lyrtech Inc. “SFF SDR DP API Guide” version 1.0, Septiembre de 2010

[3] Lyrtech Inc. “ADACMaster III User’s guide” version 1.5

[4] Texas Instrument Inc. Página web principal de Texas Instrument. Disponible en http://www.ti.com. Accedido Mayo 2012.

[5] Xilinx Inc. Página web principal de Xilinx. Disponible en http://www.xilinx.com. Accedido Mayo 2012.

[6] Lyrtech Inc. Página web principal de Lyrtech. Disponible en http://www.lyrtech.com. Accedido Mayo 2012.

[7] TMS320DM6446 Digital Media System-on-Chip Data Sheet, Texas Instruments.Septiembre de 2010.

[8] VALVERDE, Carlos. IMPLEMENTACIÓN DE UN SISTEMA OFDM EN UN DISPOSITIVO SFF SDR. Tech. rep., Universidad Carlos III de Madrid, Departamento de Teoría de la Señal y Comunicaciones, 2010

[9] FONSECA, Rony. IMPLEMENTACIÓN DE UN SISTEMA DE COMUNICACIÓN EN UN DISPOSITIVO RADIO. Tech. rep., Universidad Carlos III de Madrid, Departamento de Teoría de la Señal y Comunicaciones, 2012

[10] Virtex 4 FPGA User Guide, Xilinx, December 2008.

[11] Texas Instrument Inc. “Code Composer Studio Development Tools v3.3. Getting Started Guide”. Octubre de 2007

[12] DAC5687 de Texas Instrument. Septiembre 2006

[13] Miguel Ángel Freire Rubio “Introducción al lenguaje VHDL”. UPM-Dep. de Sistemas Electrónicos y de Control. Febrero de 2008

[14] Wikipedia. Disponible en http://es.wikipedia.org/. Septiembre de 2010.

[15] Analog Devices. Disponible en http://www.analog.com. Accedido Mayo 2012.

[16] Lucent Technologies. Disponible en http://en.wikipedia.org/wiki/Lucent. Accedido Mayo 2012.

[17] Freescale. Disponible en http://www.freescale.com. Accedido Mayo 2012.

[18] Altera. Disponible en http://www.altera.com. Accedido Mayo 2012.

[19] Using Xilinx IP Cores. Amontec, 2001.